NASA/TM—2007-214690

An Introduction to Thermodynamic Performance Analysis of Aircraft Gas Turbine Engine Cycles Using the Numerical Propulsion System Simulation Code

Scott M. Jones
Glenn Research Center, Cleveland, Ohio

March 2007

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question via the Internet to *help@sti.nasa.gov*

- Fax your question to the NASA STI Help Desk at 301–621–0134

- Telephone the NASA STI Help Desk at 301–621–0390

- Write to:
  NASA Center for AeroSpace Information (CASI)
  7115 Standard Drive
  Hanover, MD 21076–1320

NASA/TM—2007-214690

An Introduction to Thermodynamic Performance Analysis of Aircraft Gas Turbine Engine Cycles Using the Numerical Propulsion System Simulation Code

Scott M. Jones
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

March 2007

This report is a formal draft or working paper, intended to solicit comments and ideas from a technical peer group.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA Center for Aerospace Information
7115 Standard Drive
Hanover, MD 21076–1320

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161

Available electronically at http://gltrs.grc.nasa.gov

An Introduction to Thermodynamic Performance Analysis of Aircraft Gas Turbine Engine Cycles Using the Numerical Propulsion System Simulation Code

Scott M. Jones
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Abstract

This document is intended as an introduction to the analysis of gas turbine engine cycles using the Numerical Propulsion System Simulation (NPSS) code. It is assumed that the analyst has a firm understanding of fluid flow, gas dynamics, thermodynamics, and turbomachinery theory. The purpose of this paper is to provide for the novice the information necessary to begin cycle analysis using NPSS. This paper and the annotated example serve as a starting point and by no means cover the entire range of information and experience necessary for engine performance simulation. NPSS syntax is presented but for a more detailed explanation of the code the user is referred to the NPSS User Guide and Reference document (ref. 1).

The Gas Turbine Engine

The Brayton (or Joule) cycle best describes the operation of an air-breathing gas turbine engine. The engine consists of three main components—a compressor, burner, and turbine. For aircraft propulsion the cycle is open, meaning the turbine exhaust is vented to the atmosphere rather than recirculated back through the compressor. Chemical energy provided by the fuel is converted to useful work in one of two ways: the turbine can produce additional mechanical (shaft) power; or, a nozzle can provide a momentum increase in the working fluid (air) to produce thrust. Due to the complex interactions between the engine components as well as large changes in environmental operating conditions, computer codes are required to predict the performance of all but the most simple, ideal engine. NPSS is one such code and its object-oriented nature enables nearly any conceivable engine architecture to be accurately modeled.

Basic NPSS Input File Syntax

The basic format of an NPSS input file is text-based. This text can be broken down into three main categories: creation of objects, assignment of values to variables, and commands (or function calls). In object-oriented codes the creation of a particular instance of an object type is called instantiation. The input file is read (or parsed) sequentially so normally an object will be created/instantiated then values assigned to its pertinent variables and the process repeated with the next object. Once all objects have been declared then commands or function calls are made to the code. Because NPSS allows user-defined functions it is important to distinguish between *creation* of a function and actually *calling or executing* that function.

When assigning values to variables it is important to understand variable "scope" which, loosely defined, is the region where a variable exists. In order to assign a value to a variable while outside its scope, any objects above that scope will have their name prepended to the variable name with a "dot" separator (e.g., *object_1.object_2.variable_name* = 10). See the "**Variable Scope**" section of the NPSS User Guide and Reference for a more detailed explanation. Finally, remember that NPSS is case sensitive and all commands and variable assignments must end with the semicolon character.

Example Model Input Structure

Input files for NPSS may have any reasonable name or extension; a simple turbojet model consisting of several files is listed in appendix B. Appendix A is a detailed explanation of this example model. Because input files are read sequentially, models typically have a layout reflecting steps **a** through **g** outlined below:

a) Comments

Because input files may have different names, it is good practice to place a comment box at the beginning of every file explaining what that file does or the data it contains. The double slash (//) comments everything to the end of that single line, and multiple lines may be commented by using the slash and asterisk (/* … */) at each end of the comment.

b) Specify Thermodynamic Package

The thermodynamic package to be used is specified first. Many NPSS objects cannot be created unless this has been done, and this is one of the few times a command will occur near the beginning of an input file. The thermodynamic package determines fluid properties (viscosity, enthalpy, thermal conductivity, etc.) from known fluid state conditions (pressure and temperature, for example). The command to specify the thermodynamic package is **setThermoPackage("name")**; where 'name' is either 'GasTbl', 'Janaf', 'allFuel', or 'CEA'. Janaf is well suited to account for mixtures of air and water vapor typically found in a gas turbine environment. CEA is an implementation of the CEA code (ref. 2) which can account for many chemical species and interactions but at significant cost in computational time and stability. It is recommended that 'Janaf' be used unless chemical equilibrium calculations are required or another package is deemed more appropriate.

c) Physical Components and Configuration (Linkages)

Next, the model is actually configured by instantiating objects which represent the engine physical components and linking them to form the model structure. These objects are Elements and Subelements such as inlets, compressors, ducts, and turbines. Elements can be instantiated in any order, but a layout reflecting the way flow travels through the engine is recommended. Also, each Element is given a unique name by the user so that, for example, a model may have more than one compressor. This "building block" approach allows nearly any practical engine architecture, from turboshafts to recuperated turbofans, to be accurately simulated as a set of interconnected components.

Once all the Elements are declared, their outlet and inlet fluid ports are connected by the **linkPorts** command. This determines the actual layout of the engine (e.g., the compressor exit is connected to the burner entrance, the burner exit is connected to the turbine entrance, the turbine exit is connected to the nozzle entrance, etc.). Similarly, shaft ports are also connected by the **linkPorts** command to represent mechanical linkages in the engine.

The format of the command to connect Element ports is **linkPorts("element1.outlet_port", "element2.inlet_port", "link_name");** where 'link_name' is a unique name for the link.

d) Solver Objects (Independents and Dependents)

In order to attain a physically valid solution, an NPSS engine model must at a minimum obey conservation of mass, energy, and momentum; the NPSS solver ensures this when properly configured. Furthermore, the solver can also alter independent variable values (e.g., design fan pressure ratio) so that a desired condition is achieved (e.g., engine net thrust equals 10000 lb). Independents and Dependents are the most common solver objects used in NPSS. Some Elements in NPSS already have Independents and/or Dependents associated with them, and others can be created by the user. The creation of an Independent looks like this:

```
Independent my_indep_name {
        varName = "actual_variable_name";
}
```

where 'my_indep_name' is a unique user-specified name for the Independent (because there are likely to be many Independent objects in the model) and "actual_variable_name" (double quotes are necessary) is the actual *name* of the variable whose *value* will be altered when this Independent is active. Note the use of curly braces to set the scope of the variable 'varName'; an alternative is

```
Independent my_indep_name;
my_indep_name.varName = "actual_variable_name";
```

The creation of a Dependent is similar:

```
Dependent my_dep_name {
        eq_rhs = "variable_parameter_1";
        eq_lhs = "variable_parameter_2";
}
```

where 'my_dep_name' is a unique user-specified name for the Dependent (because there are likely to be many Dependent objects in the model). The variables 'eq_rhs' and 'eq_lhs' can be thought of as the right and left hand sides of an equation, or a condition to be satisfied. The strings "variable_parameter_1" and "variable_parameter_2" (double quotes are necessary) can be variable names, numerical values, or even calculations. As examples:

```
eq_rhs = "engine_thrust";
eq_lhs = "10000";

eq_rhs = "engine_thrust";
eq_lhs = "desired_thrust";

eq_rhs = "engine_thrust";
eq_lhs = "desired_thrust*2.0 – 5000.0";
```

Note that in these examples 'engine_thrust' and 'desired_thrust' must be recognized NPSS variables (although they can be user-created variables).

e) Set Up Solver (Make a Set of Independents and Dependents Active)

An engine model may contain any number of Independents and Dependents, but they are not automatically added to the solver upon their creation. Independents and Dependents are added to the solver by the **solver.addIndependent("my_indep_name")** and **solver.addDependent("my_dep_name")** commands. Conversely, active solver objects may be made inactive by the **solver.removeIndependent("my_indep_name")** and **solver.removeDependent("my_dep_name")** commands. The remove command only makes solver objects inactive, it does not delete them; they can be made active again later in the model simulation.

When the **run()** command is issued, the NPSS solver will execute the engine model to convergence if possible. This means that each Element and Subelement will perform its calculate() function; and, any and all active Independent variable values will be altered to satisfy all of the active Dependent conditions. Elements are likely to have their calculate() function executed many times as the model iterates to a solution.

Note that although any number of Independents and Dependents can be *created*, the number of *active* Independents must always equal the number of *active* Dependents for any particular case at the time of the **run()** command. Finally, the user may send to the screen a listing of the active Independents and Dependents at any point in a simulation by the following commands:
cout << solver.list("Independent", TRUE) << endl;
cout << solver.list("Dependent", TRUE) << endl;
This can be most helpful when evaluating the behavior of a model simulation.

f) Run the Case

Once the solver has been set up to provide the desired analysis, the **run()** command is issued. Data may then be written to one or more files immediately after the engine analysis case is run. A single input file will almost always run many cases; for each separate case a few variable values are changed (such as altitude and flight Mach number) and steps "e" and "f" are repeated.

g) Output Format Specification (DataViewers)

By default, no output is generated by NPSS; the user must either use the **cout** command (which sends data to standard output) or create one or more DataViewer objects which send output to a file (or an output stream). A DataViewer uniquely determines which variables to output as well as the look and format of the output. Each DataViewer object in the model is an object with a unique name and therefore the user can output data to them whenever and only as desired. To output the data to a file the command **dataviewer_name.display()** is issued, where 'dataviewer_name' is the unique name of that DataViewer. Each DataViewer also has an output stream object associated with it; this output stream specifies the actual file where the data is written (the 'filename' string variable). There are several types of DataViewer objects and their use is explained further in the NPSS User Guide and Reference.

With the exception of DataViewers, which can be specified almost anywhere in the input file as long as it is prior to their display command being issued, the creation of an NPSS input file typically follows this "recipe" of steps "a" through "f". An input file may repeat certain steps in order to analyze many cases, the first case being the design point analysis and all other cases representing off-design analysis points of interest.

Thermodynamic Performance Analysis—Design and Off-Design

Cycle analysis of air-breathing engines is done in two stages. First the engine components are sized according to the inputs specified by the user; this is called the design case or design analysis. The design case can also be considered a reference condition. Normally only one design case is executed for each model run; if multiple cases are being run then while the engine is in design mode each case results in a completely different engine.

The second stage of cycle analysis is off-design analysis. Here an engine whose design variables are known has its operating characteristics calculated for a particular flight Mach number, altitude, and throttle setting. For this reason off-design analysis is often called performance analysis. Hundreds of off-design cases are run to identify the engine operational envelope as well as any unforeseen complications resulting from potentially poor design choices. So in design mode the engine physical parameters are set, and in off-design mode that engine's performance is calculated over a range of varying conditions (flight speed, altitude, etc.).

Design Point Analysis

In order to analyze the performance of an air-breathing engine cycle, a minimum amount of input and design conditions must be specified. Table 1 is a listing of the more common engine simulation elements

and the minimum necessary input variables for each at the engine design point. Usually turbomachinery maps will be used for the performance analysis of compressors and turbines. The two different sets of inputs are (not including the map table variables) listed in the bottom half of table 1. Since turbine pressure ratio is usually a fallout from design choices rather than a design variable itself, the turbine pressure ratio scaler is determined from the desired (unscaled map input) value pressure ratio and the actual pressure ratio. See appendix C for more information on turbomachinery map scaling.

TABLE 1.—CRITICAL INPUT VARIABLES

Element type	Variable name(s)
FlightConditions	alt, MN, and W
Inlet	eRamBase (direct input or subelement)
Compressor	*see below*
Splitter	BPR
FuelStart	LHV
Burner	Wfuel/FAR/TtCombOut
Turbine	*see below*
Mixer	stream1 MN
Duct	dPqPbase (direct input or subelement)
Nozzle	PsExh or PsExhName, switchType
Shaft	Nmech, declared ShaftInputPorts
compressor variables (no map) and description	
NpctDes	percent of physical speed
eff	adiabatic efficiency
PR	pressure ratio
compressor variables (with map) and description	
alpha	stator/IGV setting or 3–D argument setting
RlineMap	the value of R-line to use @ design
NcDes	the value of corrected speed to use @ design
PRdes	the desired (scaled) value of pressure ratio
effDes	the desired (scaled) value of efficiency
turbine variables (no map) and description	
eff	adiabatic efficiency
PRbase	pressure ratio
turbine variables (with map) and description	
parmGeom	vane setting or 3–D argument setting
parmMapDes	the desired (unscaled) value of PR @ design
parmNcDes	the value of corrected speed to use @ design
parmMap	the actual (scaled) value of pressure ratio
effDes	the desired (scaled) value of efficiency

At the design point, NPSS automatically balances and matches most of the engine by doing the following: a) changing the scale factors on the map corrected flows for all compressors and turbines so that each is correctly sized to pass the flow into that component; b) setting the exit area (and if applicable, throat area) for all nozzles based on the nozzle entrance flow and back pressure; c) setting the area of ducts and flowstations (if Mach number is provided); and d) setting areas for mixer inlet streams by matching static pressures and setting mixer outlet total pressure based on conservation of momentum (zero net impulse) assuming a constant area mixer. All Elements in NPSS have an option variable called "switchDes" which can be set to either "DESIGN" or "OFFDESIGN"; this variable determines whether certain variables are input or output (i.e., calculated).

What normally remains at design is an energy balance for each shaft. The user must add solver objects that vary the expansion ratio across each turbine (variable 'parmMap') to match the supplied turbine work (or torque) to each shaft with the torque required by all compressors and loads also on that shaft (variable 'trqNet'); these solver objects will be added automatically if the **autoSolverSetup()** command is issued at design. Finally, other design conditions can be included if desired. Examples are

varying compressor pressure ratio to achieve a certain thrust specific fuel consumption (TSFC) and varying engine airflow to meet a certain net thrust.

Off-Design Point Analysis (Steady State)

In NPSS, Independent and Dependent objects are used to balance the engine by varying x so that y is satisfied, where x is the free variable specified by the Independent and y is the mismatch we want to balance specified by the Dependent. Because this is a system of simultaneous equations the number of active Independents (variables) must always equal the number of active Dependents (equations) for each case, although the number may vary from case to case.

At off-design there naturally exists a flow mismatch at the entrance to any nozzle or rotating component. In addition, there is a shaft power mismatch between the turbine-delivered power and the compressor-required power. The engine is balanced at off-design by altering the independent (or free) variables of available components to eliminate these discontinuities.

Table 2 lists the most common free variables available to the user at off-design while table 3 lists component errors that need to be resolved for a physically valid solution. So for a two-shaft separate flow turbofan each rotating component's free variable is used to balance the flow mismatch at the entrance to that particular component and the shaft rpms are used to balance the power delivered/required by each shaft. That leaves inlet airflow and bypass ratio to balance the flow mismatch at the two nozzle entrances. Any remaining independents are available for the user to control the engine; this is explained further in the next section.

TABLE 2.—AVAILABLE INDEPENDENT VARIABLES AND CORRESPONDING INDEPENDENT OBJECT NAMES

Element type	Variable name	Independent name
FlightConditions	W	ind_W
CompressorMap(subelement)	RlineMap	ind_RlineMap
TurbineNeppMap(subelement)	parmMap	ind_parmMap
Shaft	Nmech	ind_Nmech
Splitter	BPR	ind_BPR
Burner (switchBurn == WFUEL) (switchBurn == TEMPERATURE) (switchBurn == FAR	 Wfuel, or TtCombOut, or FAR	 must be created by user must be created by user must be created by user
FlowStart	W Pt Tt	must be created by user must be created by user must be created by user
Mixer	none—(internal to mixer)	----------
Nozzle	none—(fixed area) AthCold—(variable area)	---------- must be created by user

TABLE 3.—SYSTEM DISCONTINUITY ERRORS AND DEPENDENT NAMES

Element type	Error/mismatch	Dependent name
CompressorMap	flow into the compressor compressor map flow based on Rline and Nc	dep_errWc
TurbineNeppMap	flow into the turbine turbine map flow based on PR and Nc	dep_errWp
Nozzle	flow into the nozzle nozzle flow based on area and PR	dep_Area
Shaft	torque/work required for all compressors and loads torque/work delivered by all turbines	integrate_Nmech
Mixer	(1) stream 1 static pressure (1) stream 2 static pressure (2) stream 1 impulse + stream 2 impulse (2) exit stream impulse * loss factor	dep_errPs none—(internal)

Off-Design Full Power Operation (Maximum Power)

In the case of the two-shaft separate flow turbofan mentioned above one free variable remains, burner fuel flow, to be used as desired. If burner fuel flow is not explicitly changed or made active in the solver it will remain constant. If the engine is operated over a flight envelope this way the turbomachinery (mostly the fan and LPC) is forced to odd operating points (e.g., fan at extremely high corrected speed, LPC at low surge margin): a way to determine and control fuel flow as engine operating conditions change is needed.

For off-design analysis it is not sufficient to merely balance engine discontinuities, a definition of full power operation is required. Examples of full power definitions are maintaining fan corrected speed at 100 percent, maintaining low spool rpm at 100 percent, or maintaining a constant high pressure turbine entrance temperature. Once the engine full power operating requirements are set, fuel flow can be calculated and the analyst can begin to see the effects of design choices on engine performance.

Off-Design Part Power Operation (Throttle Curves)

This is a subset of off-design operation. While an engine may operate at maximum power during an aircraft takeoff and climb, it is necessary to throttle back the engine during long cruise segments and especially during descent. The essence of part power operation is simply a matter of decreasing burner exit temperature from the defined maximum at some Mach number and altitude and letting NPSS balance the engine. There are several ways of doing this, each with advantages and disadvantages, but all yield the same results. Three methods of defining part power operation are explained below.

The first method is to directly input burner exit conditions. Simply run a series of cases with the burner exit temperature/fuel flow/FAR input (which one is input depends on the value of the burner variable ‘switchBurn’) and decreasing from case to case. This method is simple but it requires prior knowledge of the starting (full power) burner temperature/fuel flow/FAR which may or may not be constant from case to case.

The second method is to vary burner exit conditions to yield a given net thrust. To do this an Independent and Dependent must be created and made active in the solver. The Independent variable comes from the burner Element and depends on the value of the ‘switchBurn’ variable. The Dependent will equate the engine thrust performance variable to a target thrust value. The target thrust value is frequently defined in terms of a percentage of the engine maximum thrust at each flight condition for convenience.

The third method is to vary burner exit conditions to yield a given fan corrected speed (or high pressure compressor corrected speed for turbojets). This is similar to the second method; only the Dependent conditions are different.

Common Error Messages in NPSS: What They Mean and How to Fix Them

NPSS has a large array of warning and error messages to assist the user in eliminating improper conditions in a model simulation. When ‘debugging’ a model it is critical that the novice user remedy the first error message only! It is very common for a single input error to cascade into a screen-scrolling nightmare. The following are some of the more common error messages that occur when constructing a model simulation and what causes them.

ERROR(71001001) in file *filename* – line *n* – Unresolved variable ‘*xxx*’
All variables in NPSS must be explicitly declared (as real, int, or string); this error means NPSS has encountered a variable which has not been declared. The variable in question could be user-created and this error has four likely causes:

1) the variable name is misspelled or incorrectly capitalized (this is the likely cause when the variable is not user-created)
2) the variable is simply not declared at all (for user-created variables)
3) the variable is declared but after NPSS first encounters it being assigned a value
4) the variable is declared but referenced outside its scope (make sure the scoping name is correctly spelled)

WARNING(71021001) in file *filename* – line *n* – in Solver 'solver': VERIFY FAILED : Jacobian indep and dep lists are not the same length (#indep = *n1*, #deps = *n2*)
This warning indicates the number of active Independents, *n1*, is not equal to the number of active Dependents, *n2*, at the time of the **run()** command. An error message will follow this warning.

ERROR(81023999) in file *filename* – line *n* – in SecantSolver '*xxx*.smSolver' : Slope between two points went off to infinity:
For this error message '*xxx*' is likely to be a rotating component name and the possible cause is that the component operating point has moved beyond its map boundaries. Alternatively, if the mechanical speed of a shaft (*shaft_name*.Nmech) connected to a compressor is inadvertently set to zero at design, this error will result.

ERROR(81031001) in file *filename* – line *n* – in Assembly '': function *xxx*_version was not found in lib *xxx*.dll
Due to the case sensitivity of NPSS contrasted with the case insensitivity of some computer file systems, this error is probably the result of not capitalizing the first letter of an Element type like "compressor" instead of "Compressor".

ERROR(91003001) in file *filename* – line *n* – in MemberFunction 'run': Exception caught in SolverExecutive::run : Exception caught in SolverExecutive::convergeContinuous
Model : Exception thrown from matrix generation step : Exception caught in Jacobian::generateMatrix Exception caught during matrix inversion Exception caught in Jacobian::determineColumnNormalizerSingular Matrix - zero column: Perturbing matrix-independent number #*n1*, *independent_name*, did not effect any of the error terms
The last part of this long error message is what matters. Each active Independent variable must have an effect on at least one active Dependent term. Note that if the dependency is extremely small (but not exactly zero) an unconverged solution error will result instead. Assuming that the named Independent can significantly affect at least one Dependent then there are two likely causes:

1) the independent variable is repeatedly being assigned values somewhere other than just the solver, as in a **preexecute()** function
2) the independent variable value is beyond the range of a table or function whose extrapolation is set to "NONE", resulting in the same tabular value being returned regardless of the independent value

ERROR(91003001) in file *filename* – line *n* – in MemberFunction 'run': Exception caught in SolverExecutive::run : Exception caught in SolverExecutive::convergeContinuous
Model : Exception thrown from matrix generation step : Exception caught in Jacobian::generateMatrix Exception caught during matrix inversion Exception caught in Jacobian::determineRowNormalizer Singular Matrix - zero row: The error for matrix-row number *n1*, *dependent_name*, is not affected by any of the independent perturbations
Similar to the previous error, this error occurs when a dependent term remains unchanged by any Independent variable change. Two likely causes:

1) the Dependent variable terms (“eq_rhs”, “eq_lhs”, or both) are not suitably linked to the active Independents
2) the Dependent variable terms (both “eq_rhs” and “eq_lhs”) are ultimately being assigned to constants (e.g., vary *x* so that 5 equals 7 is impossible)

ERROR(91003001) in file *filename* – line *n* – in MemberFunction 'run': Exception caught in SolverExecutive::run : Exception caught in SolverExecutive::convergeContinuous Model : Exception thrown from matrix generation step : Exception caught in Jacobian::generateMatrix Exception caught during matrix inversion Exception caught in Jacobian::determineRowNormalizer Linearly Dependent Matrix: The coefficients for matrix-row *#n1*, *dependent_name1*, are a scalar multiple of those for matrix-row *#n2*, *dependent_name2*
This error means that two of the active Dependents are really the same. For example, if x and y are variables the two equations x+y = 5 and 2x+2y = 10 can not each be used as Dependent conditions since they are trivially related. Another possibility is when the same variable can be referenced by different names; an example would be a burner connected to a turbine: the same temperature variable can be called *burner_name*.Fl_O.Tt and *turbine_name*.Fl_I.Tt.

ERROR(91003201) in file *filename* – line *n* – in MemberFunction ‘run’: Exception caught in SolverExecutive::run : Solver FAILED TO CONVERGE continuous model in *m* iterations.
Unfortunately, this error is as common as it is difficult to determine its exact cause. It occurs when the solver is unable to find a solution within the given tolerance and within the number of allowed iterations. Making matters worse is this is not a fatal error, allowing model execution to continue and data to be output to files where there may be no indication that it is invalid. General things to look for when this error is encountered:

1) typically increasing the allowable number of iterations beyond 100 will <u>not</u> solve the problem
2) when the design point fails to converge, check the initial guesses and values for the Independent variables to make sure they are at least the same order of magnitude as the final values—if airflow starts at 0.1 lb_m/s, it will take too many perturbations to reach 100 lb_m/s
3) also at design, make sure each Dependent is significantly affected by at least one Independent - varying HPC design adiabatic efficiency to achieve a certain HPT design polytropic efficiency is an example of a minuscule dependence (HPC efficiency affects HPT pressure ratio which affects HPT polytropic efficiency)
4) for off-design, keep changes in flight Mach and altitude relatively small from case to case—going from sea level, static conditions to Mach 0.85, 40000 ft altitude is fine for a simple turbojet but far too big a change for complex engine simulations (usually $\Delta MN \approx 0.10$ and $\Delta h \approx 2000$ ft are acceptable)
5) verify that Independent variables are not going beyond table ranges and maps which can result in odd extrapolation (or no extrapolation)
6) occasionally a case will not converge because the model inputs are just right to cause the solver to consistently overshoot the solution – if the engine simulation works both at 9990 ft and 10010 ft altitudes but not at 10000 ft then this is the likely problem

ERROR(91041001) in file *filename* – line *n* – in Assembly ‘’: Cannot create ‘*my_object_name*’ of type ‘*xxx*’
Similar to the unresolved variable error, this error arises when the user tries to instantiate an object that NPSS does not recognize. Likely causes are:

1) the object type, ‘*xxx*’, is misspelled like “Nozzel” or “Indepenent”
2) the object type should be capitalized like “Independent”, not “independent”

ERROR(91041001) in file *filename* – line *n* – Could not create FlowStation because you haven't specified a thermo package.
NPSS needs a thermodynamic package in order to create some of its objects; make sure the **setThermoPackage("xxx")** command is one of the first things in the input file.

Appendix A
Annotation of Turbojet Example Model

In the turbojet example, shown schematically in figure 1, the model is composed of eleven Elements, seven of which represent the physical components inlet, compressor, burner, turbine, duct, nozzle, and a shaft connecting the turbomachinery. The four remaining Elements do not represent physical machinery but nevertheless provide important data, set certain conditions, or perform important functions. These four Elements are the following:

FlightConditions—this Element calculates atmospheric properties and starts the fluid flow stream
FuelStart—this Element specifies fuel properties
FlowEnd—this Element is required to end the fluid flow stream
EngPerf—this is an optional Element that calculates engine performance variables

In general, the minimally required or most important variables for each Element are assigned values. In instantiation order, the eleven Elements in the turbojet model are:

1) FlightConditions Element named "Ambient"

This Element calculates atmospheric properties based on the two input variables 'alt', the engine altitude in feet, and 'MN', the flight Mach number of the engine. Here the example turbojet is operating at sea level (altitude = 0) static (velocity or MN = 0) conditions. The FlightConditions Element also specifies the amount of air mass flow into the engine (W = 100 lb_m/s).

2) Inlet Element named "Inlet"

This Element adjusts the fluid pressure according to the inlet stagnation pressure recovery variable, 'eRamBase'. In the turbojet example the inlet pressure recovery is 0.980, or a 2 percent total pressure loss across the inlet component.

3) Compressor Element named "HPC"

This Element is used to represent high pressure compressors, low pressure compressors, and fan components in engine models. Because of the large amount of text and information in all but the simplest of models, NPSS input is often contained in multiple files for easier study. Any input file can be included by the statement **#include<my_file_name>**, where 'my_file_name' is the name of the file (double

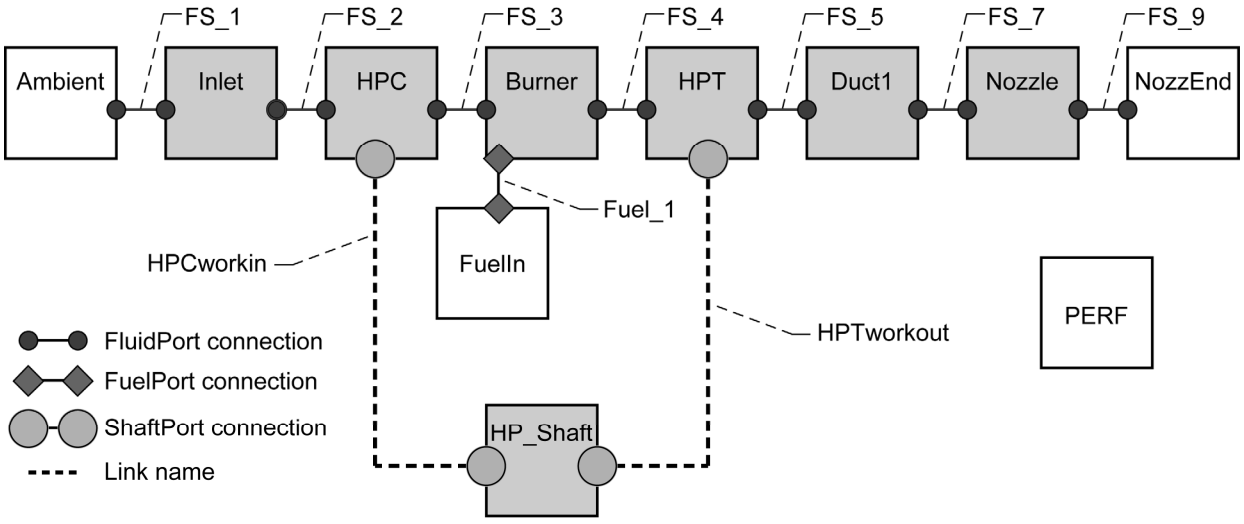

Figure 1.—Example turbojet NPSS thermodynamic cycle model.

quotes in place of angle brackets may also be used). In the example model, the compressor has a CompressorMap Subelement called ‘S_map’ which contains three data tables; this Subelement and associated tables are contained in the “turbojet_HPC.map” file which is included on line 35 of the model file. See appendix C for a discussion of how turbomachinery maps are used in NPSS. In the turbojet model the compressor has a pressure ratio of 20.0 and an adiabatic efficiency of 85.0 percent.

4) FuelStart Element named “FuelIn”

The purpose of the FuelStart Element is to specify the properties of the fuel that will be added and burned in the combustor. The default fuel is JP–4, and the most common input variable is the fuel lower heating value, or LHV. The fuel in the example model has an LHV of 18400 BTU/lb_m. This Element can also be considered as starting a flow stream of fuel, although the flow rate is usually set in the Burner Element.

5) Burner Element named “Burner”

The Burner Element adds fuel to an airflow and burns it, resulting in a temperature rise. The Burner calculations depend on the ‘switchBurn’ variable: if ‘switchBurn’ = “TEMPERATURE”, the Burner exit temperature is set equal to the value of the variable ‘TtCombOut’ and the necessary fuel-to-air ratio (FAR) and fuel flow are calculated. If ‘switchBurn’ = “FUEL”, the Burner adds fuel equal to the value of the variable ‘Wfuel’ and the corresponding fuel-to-air ratio (FAR) and exit temperature are calculated. If ‘switchBurn’ = “FAR”, the Burner exit fuel-to-air ratio is set equal to the value of the variable ‘FAR’ and the corresponding fuel flow (Wfuel) and exit temperature are calculated. This last method is used in the turbojet example to set the Burner exit fuel-to-air-ratio to a value of 0.030. Recalling that the input airflow is 100 lb_m/s, the fuel flow for this model will be 3.0 lb_m/s.

The Burner also has a stagnation pressure drop specified by the variable ‘dPqPBase’ (i.e., $\Delta P/P$). The turbojet example has a stagnation pressure loss of 5 percent across the Burner relative to the Burner entrance stagnation pressure.

6) Turbine Element named “HPT”

This Element is used to represent both high and low pressure turbine components. Like the compressor, the turbine in the example model has Subelements and tabular data contained in an included file, “turbojet_HPT.map”. See appendix C for more information on turbine maps. The example model turbine has an adiabatic efficiency of 92.0 percent.

7) Duct Element named “Duct1”

This Element is primarily used to simulate pressure drops from frictional effects in long passages or losses from turning the fluid flow. The pressure drop can be set from the input value of ‘dPqPbase’, a function of duct entrance Mach number, or a user-specified loss function. The example model has a stagnation pressure loss of 1.0 percent.

8) Nozzle Element named “Nozzle”

The Nozzle Element represents both convergent and convergent-divergent nozzles as determined by the ‘switchType’ variable. Thrust calculations are based on either velocity coefficient or gross thrust coefficient as determined by the value of the ‘switchCoef’ variable. To do the nozzle calculations an exhaust or back pressure is required; this can be done by either input of the back pressure itself (variable ‘PsExh’) or a link to the proper pressure (variable ‘PsExhName’). Normally the nozzle exhausts to the atmosphere and this can be specified by setting ‘PsExhName’ equal to the model’s FlightConditions Element static pressure variable, as is done in the example model. The example nozzle is convergent (‘switchType’ = “CONIC”) with losses based on velocity coefficient (‘switchCoef’ = “CV” and ‘Cv’ = 0.98).

9) FlowEnd Element named “NozzEnd”

This Element is used to terminate a flow stream; all flow streams in NPSS must have a beginning (either FlowStart or FlightConditions Element) and an end.

10) Shaft Element named “HP_Shaft”

The Shaft Element is used when a mechanical link exists between two or more rotating components. Unlike many Elements which are connected by their FluidPorts, this Element connects to others via ShaftPorts. By default the Shaft Element has no ShaftPorts; instead its ShaftPorts are declared by the user so that it can have any number appropriate to the engine being simulated. The command to instantiate a ShaftPort is **ShaftInputPort name**, where ‘name’ is a unique name for that port. In the turbojet example the shaft element has two ShaftInputPorts called ‘Sh_HPC’ and ‘Sh_HPT’; multiple ports can be instantiated in one command using commas to separate the names as in the example. Once the ports have been declared, they can be linked; these ports will be connected to the ShaftOutputPorts on the model’s compressor and turbine later in the input file.

The other thing of note in the Shaft Element is the ‘Nmech’ variable which is the shaft rotational speed in rpm; the example model has a shaft speed of 10000 rpm.

11) EngPerf Element named “PERF”

By design, NPSS models can be very different: a model may represent a turboprop, turbofan, just the core of an engine, etc. Because of this, a priori equations and variables related to overall engine performance for every conceivable model configuration are impossible. However, the EngPerf Element calculates some common overall engine performance variables such as net thrust and thrust specific fuel consumption. This Element assumes that the model is a thrust-producing configuration (not a turboshaft or turboprop) with at least one inlet, at least one burner, and at least one nozzle. In addition, these Elements must be the ‘standard’ NPSS release Elements, not user-created Elements. The turbojet example can use the EngPerf Element. In the event that the EngPerf Element is not applicable, the user has three choices:

1) omit it and not have any performance variables,
2) omit it but add their own performance calculations somewhere else, or
3) modify it to correctly calculate their performance variables

After all the Elements in the model have been declared they must be linked together forming a coherent simulation of a gas turbine engine. The types of ports used in NPSS are FluidPorts, ShaftPorts, and ThermalPorts. Ports are linked using the **linkPorts** command:
linkPorts(“element1.outlet_port”, “element2.inlet_port”, “link_name”);
where ‘link_name’ is a user-specified, unique name for the link. By convention, most FluidOutletPorts are called “Fl_O”, FluidInputPorts are called “Fl_I”, and ShaftOutputPorts are “Sh_O”. Finally, FluidOutputPorts can only be linked to FluidInputPorts; likewise, ShaftOutputPorts can only be linked to ShaftInputPorts. In the example model there is a FuelOutputPort on the FuelStart Element connected to the FuelInputPort on the Burner Element, and the two declared ShaftInputPorts on the Shaft Element (called “Sh_HPC” and “Sh_HPT”) are connected to the ShaftOutputPorts on the Compressor and Turbine Elements respectively.

Once all necessary Elements are instantiated, values given to important variables, and all of the Element ports are linked, the model simulation can be run. This consists of any necessary changes to the solver followed by the **run()** command. The **autoSolverSetup()** command makes all Independent and Dependent objects inactive then activates any Independent or Dependent whose “autoSetup” variable is set to TRUE.

In the turbojet example, the command **setOption(“switchDes”, “DESIGN”)** sets the value of every variable called “switchDes” in the model to “DESIGN”. A consequence of this is that certain Independents and Dependents have their “autoSetup” variable value set to either TRUE or FALSE. In this

case the HPT map subelement will have its Independent, “ind_parmMap”, added to the solver and the HP_Shaft Element will have its Dependent, “integrate_Nmech”, added to the solver when the **autoSolverSetup()** command is issued because their “autoSetup” value is TRUE. In this example no other Independent or Dependent is active during the design case run.

After the **run()** command, the **cout** command is used to display to the screen simple text messages and variables related to engine performance. A DataViewer declaration follows, which defines a specific type of output format and variables to be shown. In this particular DataViewer, the viewer subtype is a PageViewer and the actual name of the viewer is “page”. The name of the file this DataViewer writes to is called “turbojet.output”. Declaration of this DataViewer does not send any data to the file; that is done when the DataViewer’s **display()** command is given, as seen on the next line in the example model. Notice that the **display()** command is a function of each individual DataViewer, so it must be given proper reference, or scope, by first giving the name of the particular DataViewer whose output is desired. In this case the DataViewer name is “page”, hence **page.display()**.

Next, two off-design points are run. The command **setOption(“switchDes”, “OFFDESIGN”)** is issued to set each element in off-design mode and update the values of the “autoSetup” variables. The **autoSolverSetup()** command now activates the following solver objects: the HPC map Independent, the HPT map Independent, the shaft Independent, the engine airflow Independent, the HPC map Dependent, the HPT map Dependent, the shaft Dependent, and the nozzle Dependent. To summarize, the active solver objects are:

active Independents	active Dependents
“HPC.S_map.ind_RlineMap”	“HPC.S_map.dep_errWc”
“HPT.S_map.ind_parmMap”	“HPT.S_map.dep_errWp”
“HP_Shaft.ind_Nmech”	“HP_Shaft.integrate_Nmech”
“Ambient.ind_W”	“Nozzle.dep_Area”

The case is run and data sent to the PageViewer output file. Since the engine operating conditions were not changed, the output for this off-design point, case “a”, will exactly match that for the engine’s design point. However, for the second off-design point, case “b”, the engine operating conditions are desired at a flight Mach number of 1.00 and a flight altitude of 20000 ft. In addition, the burner exit fuel-to-air ratio is changed to 0.0314. If the burner FAR is not explicitly changed by the user (or associated with an active Independent), it will remain constant. As before, the case is run and the data output.

As a final example, another off-design point is run with a set of user-defined solver objects active in addition to the others. In the previous case, the burner fuel-to-air ratio was set at 0.0314, yielding a combustor exhaust temperature (T_4) of approximately 3160 Rankine. Suppose a T_4 equal to 3200 Rankine is desired: this could be achieved by trial-and-error input of burner FAR, but creation and use of an Independent and Dependent is far superior. Case “c” illustrates this process and runs the case; it can be seen from the output that burner FAR has been altered to produce a T_4 of exactly 3200, within solver tolerance. Once created, these Independent and Dependents do not have to be specified again. It is advisable to create all Independents and Dependents in the same place and use them as necessary for each case remembering that they always remain in the same state (whether active or inactive) from case to case unless explicitly changed.

Appendix B
Example Turbojet File Listing

The example turbojet model consists of the following five files:

turbojet.mdl	this file configures the engine and runs the simulation
turbojet_HPC.map	this file contains the compressor performance map
turbojet_HPT.map	this file contains the turbine performance map
turbojet.view_page	this file determines the format of the output
turbojet.output	this is the output file generated by running the simulation

Again, file names can be completely arbitrary but these were chosen based on convention to reflect their contents. The "turbojet.view_page" file contains a complex DataViewer specification which uses NPSS functions to generate an output format compatible with most model simulations that use the standard NPSS Elements. This DataViewer will not be explained in detail, but several lines of interest are explained. First, an output stream name must be created with an associated filename where the data will be written:

```
OutFileStream pageStream { filename = "turbojet.output"; }
```

Here the output stream is called 'pageStream' and all data in this stream will go to the file "turbojet.output". Note this is done outside of the actual DataViewer specification, which appears next:

```
DataViewer PageViewer page {
   // many lines which list desired output variables in specific groups, and their format
outStreamHandle = "pageStream";
}
```

This DataViewer is a PageViewer subtype (VarDumpViewer, CaseColumnViewer, and CaseRowViewer are the others) and 'page' is the user-given name for this particular viewer. The 'outStreamHandle' string variable is set equal to the OutFileStream object name previously defined ('pageStream'), thereby associating this viewer's output with that output stream and the file, "turbojet.output". Each time this DataViewer's **display()** function is called, model simulation data is written. In the example model, the "turbojet.output" file contains the engine data at four conditions: the DESIGN case and the three OFF-DESIGN cases.

```
//------------------------------------------------------------------------
//
//   File Name:     turbojet.mdl
//   Date(s):       December 1, 2005
//   Author:        Scott M. Jones
//
//   Description:   simple NPSS turbojet model for pedagogical use
//
//
//------------------------------------------------------------------------


setThermoPackage("Janaf");

/*
  this is an example of
  multiple line comments
  using the slash and asterisk
*/

Element FlightConditions Ambient {
   MN = 0.00;
   alt = 0.;
   W = 100.0;
}


Element Inlet Inlet {
   eRamBase = 0.980;   // comment: this is ram recovery
}


Element Compressor HPC {
   // use a map for the compressor from the file, "turbojet_HPC.map"
   #include <turbojet_HPC.map>
   S_map.RlineMap = 2.0000;
   S_map.NcDes = 1.000;
   S_map.PRdes = 20.00;
   S_map.effDes = 0.8500;
}


Element FuelStart FuelIn {
   LHV = 18400.;       // this is in BTU per lb mass
}


Element Burner Burner {
   dPqPBase = 0.050;
   switchBurn = "FAR";
   FAR = 0.0300;
}


Element Turbine HPT {
   // use a map for the turbine from the file, "turbojet_HPT.map"
   #include <turbojet_HPT.map>
   S_map.effDes = 0.9200;
   S_map.parmMap = 3.0;
   S_map.parmGeomDes  = 1.0;
   S_map.parmMapDes = 3.000;
   S_map.parmNcDes = 100.0;
}
```

```
Element Duct Duct1 {
   dPqPbase = 0.010;
}


Element Nozzle Nozzle {
   switchType = "CONIC";
   switchCoef = "CV";
   Cv = 0.9800;
   PsExhName = "Ambient.Ps";
}


Element FlowEnd NozzEnd {  }


Element Shaft HP_Shaft {
   ShaftInputPort Sh_HPC, Sh_HPT;
   Nmech = 10000;
}


Element EngPerf PERF {
}


//--------------------------------------------------------------------------
//                          COMPONENT LINKAGES
//--------------------------------------------------------------------------
linkPorts( "Ambient.Fl_O"    , "Inlet.Fl_I"       , "FS_1"  ) ;
linkPorts( "Inlet.Fl_O"      , "HPC.Fl_I"         , "FS_2"  ) ;
linkPorts( "HPC.Fl_O"        , "Burner.Fl_I"      , "FS_3"  ) ;
linkPorts( "Burner.Fl_O"     , "HPT.Fl_I"         , "FS_4"  ) ;
linkPorts( "HPT.Fl_O"        , "Duct1.Fl_I"       , "FS_5"  ) ;
linkPorts( "Duct1.Fl_O"      , "Nozzle.Fl_I"      , "FS_7"  ) ;
linkPorts( "Nozzle.Fl_O"     , "NozzEnd.Fl_I"     , "FS_9"  ) ;

linkPorts( "FuelIn.Fu_O"     , "Burner.Fu_I"      , "Fuel_1") ;
linkPorts( "HPC.Sh_O"        , "HP_Shaft.Sh_HPC"  , "HPCworkin"   ) ;
linkPorts( "HPT.Sh_O"        , "HP_Shaft.Sh_HPT"  , "HPTworkout"  ) ;


//--------------------------------------------------------------------------
//                       RUN THE MODEL DESIGN CASE
//--------------------------------------------------------------------------
setOption( "switchDes", "DESIGN" );
autoSolverSetup();
run();

// WRITE SOME TEXT AND DATA TO THE SCREEN
cout << "Turbojet Thrust = " << PERF.Fn << " lbs " << endl;
cout << "Turbojet Fuel Flow = " << PERF.wfuel << " lbs/hr " << endl;
cout << "Turbojet TSFC = " << PERF.TSFC << " lb/hr/lb " << endl;


//--------------------------------------------------------------------------
//                          PRINT OUT THE DATA
//--------------------------------------------------------------------------
#include <turbojet.view_page>
page.display();
```

```
//-------------------------------------------------------------------------
//                       RUN AN OFF-DESIGN CASE
//           a) AT THE SAME CONDITIONS AS THE DESIGN CASE
//-------------------------------------------------------------------------
setOption( "switchDes", "OFFDESIGN" );
autoSolverSetup();
run();
page.display();


//-------------------------------------------------------------------------
//                       RUN AN OFF-DESIGN CASE
//           b) AT A DIFFERENT ALTITUDE AND MACH NUMBER
//-------------------------------------------------------------------------
Ambient.alt = 20000.;
Ambient.MN = 1.00;
Burner.FAR = 0.0314;
run();
page.display();


//-------------------------------------------------------------------------
//                       RUN AN OFF-DESIGN CASE
//           c) USING NEWLY-CREATED SOLVER OBJECTS AND MAKING THEM ACTIVE
//-------------------------------------------------------------------------

Independent Burner_FAR {    // <-- the Independent's name is "Burner_FAR"
   varName = "Burner.FAR";  // <-- the variable that this Indep. can vary
}                           // this Independent has now been CREATED

Dependent Target_T4 {         // <-- the Dependent's name is "Target_T4"
   eq_rhs = "Burner.Fl_O.Tt"; // <-- right-hand-side of the Dep. condition
   eq_lhs = "3200.0";         // <-- left-hand-side of the Dep. condition
}                             // this Dependent has now been CREATED

solver.addIndependent( "Burner_FAR" );  // this Independent is now ACTIVE
solver.addDependent( "Target_T4" );     // this Dependent is now ACTIVE

run();
page.display();
```

```
//-------------------------------------------------------------------------
// File Name:     turbojet_HPC.map
// This data Map is generated from an existing NEPP map.
// The original NEPP map tables were:
//   3001    BASELINE HP COMPRESSOR FLOW VS. R, SPEED, AND ANGL
//   3002    BASELINE HP COMPRESSOR EFF VS. R, SPEED, AND ANGL
//   3003    BASELINE HP COMPRESSOR PR VS. R, SPEED, AND ANGL
//
//-------------------------------------------------------------------------


//  ------------------------------------------------------------------------
//  |    Declaration of a new subelement instance compressorMap
//  |    that is of the type: CompressorMap
//  ------------------------------------------------------------------------


Subelement CompressorMap S_map {
   alpha = 0.0;
   a_alpha = 0.0;
   s_alpha = 1.0;
   RlineMap = 2.000;
   NcDes = 1.000;
   PRdes = 23.00;
   effDes = 0.8520;
   ReDes = 5000000.0;
   RtMap = 0.0685512;
   gamtMap = 1.40052;

   SMWmap = 15.0;
   SMNmap = 18.0;

//  ------------------------------------------------------------------------
//  |    Declaration of a new subelement instance compressorEfficiencyMap
//  |    that is of the type: CompressorEfficiencyMap
//  ------------------------------------------------------------------------


Subelement CompressorEfficiencyMap S_eff {
 RlineStall= 1.0;

 Table TB_Wc(real alphaMap, real NcorrMap, real RlineMap) {
   alphaMap= 0.000 {
     NcorrMap= 0.500 {
       RlineMap = {    1.000,    1.200,    1.400,    1.600,    1.800,    2.000,
                       2.200,    2.400,    2.600,    2.800,    3.000 }
       WcorrMap = {  22.7411,  24.0487,  25.1548,  26.0615,  26.7738,  27.2992,
                     27.6470,  27.8286,  27.8634,  27.8634,  27.8634 }
     }
     NcorrMap= 0.600 {
       RlineMap = *;
       WcorrMap = {  31.7548,  33.1181,  34.2670,  35.2054,  35.9397,  36.4783,
                     36.8308,  37.0085,  37.0362,  37.0362,  37.0362 }
     }
     NcorrMap= 0.700 {
       RlineMap = *;
       WcorrMap = {  46.1066,  47.4088,  48.5066,  49.4046,  50.1096,  50.6291,
                     50.9717,  51.1469,  51.1757,  51.1757,  51.1757 }
     }
     NcorrMap= 0.750 {
       RlineMap = *;
       WcorrMap = {  56.7268,  58.0480,  59.1608,  60.0704,  60.7837,  61.3084,
                     61.6527,  61.8260,  61.8517,  61.8517,  61.8517 }
```

```
  }
  NcorrMap= 0.800 {
    RlineMap = *;
    WcorrMap = {  70.1448,  71.5163,  72.6688,  73.6088,  74.3429,  74.8795,
                  75.2269,  75.3943,  75.4134,  75.4134,  75.4134 }
  }
  NcorrMap= 0.850 {
    RlineMap = *;
    WcorrMap = {  89.3764,  90.9746,  92.3098,  93.3900,  94.2232,  94.8199,
                  95.1897,  95.3442,  95.3504,  95.3504,  95.3504 }
  }
  NcorrMap= 0.900 {
    RlineMap = *;
    WcorrMap = { 118.0620, 120.1207, 121.8253, 123.1867, 124.2166, 124.9292,
                 125.3385, 125.4609, 125.4609, 125.4609, 125.4609 }
  }
  NcorrMap= 0.925 {
    RlineMap = *;
    WcorrMap = { 138.5093, 140.8966, 142.8639, 144.4238, 145.5916, 146.3836,
                 146.8174, 146.9192, 146.9192, 146.9192, 146.9192 }
  }
  NcorrMap= 0.950 {
    RlineMap = *;
    WcorrMap = { 160.6243, 162.5676, 164.1805, 165.4722, 166.4536, 167.1370,
                 167.5334, 167.6563, 167.6563, 167.6563, 167.6563 }
  }
  NcorrMap= 0.975 {
    RlineMap = *;
    WcorrMap = { 181.7993, 183.4993, 184.9150, 186.0545, 186.9260, 187.5389,
                 187.9029, 188.0273, 188.0271, 188.0271, 188.0271 }
  }
  NcorrMap= 1.000 {
    RlineMap = *;
    WcorrMap = { 202.6315, 203.5858, 204.3958, 205.0661, 205.5998, 206.0000,
                 206.2702, 206.4145, 206.4418, 206.4418, 206.4418 }
  }
  NcorrMap= 1.025 {
    RlineMap = *;
    WcorrMap = { 209.9986, 210.5917, 211.1029, 211.5321, 211.8825, 212.1554,
                 212.3516, 212.4735, 212.5220, 212.5227, 212.5227 }
  }
  NcorrMap= 1.050 {
    RlineMap = *;
    WcorrMap = { 216.6847, 217.0279, 217.3287, 217.5860, 217.8015, 217.9767,
                 218.1106, 218.2041, 218.2586, 218.2739, 218.2739 }
  }
}
alphaMap= 90.000 {
  NcorrMap= 0.500 {
    RlineMap = { 1.000, 1.200, 1.400, 1.600, 1.800, 2.000, 2.200,
          2.400, 2.600, 2.800, 3.000 }
    WcorrMap = {  45.5139,  47.7923,  49.6744,  51.1682,  52.2877,  53.0508,
                  53.4791,  53.5986,  53.5986,  53.5986,  53.5986 }
  }
  NcorrMap= 0.600 {
    RlineMap = *;
    WcorrMap = {  61.9648,  64.2862,  66.1998,  67.7166,  68.8513,  69.6225,
                  70.0509,  70.1629,  70.1629,  70.1629,  70.1629 }
  }
  NcorrMap= 0.700 {
    RlineMap = *;
    WcorrMap = {  84.8030,  86.9149,  88.6608,  90.0516,  91.1005,  91.8227,
                  92.2351,  92.3561,  92.3561,  92.3561,  92.3561 }
```

```
    }
    NcorrMap= 0.750 {
      RlineMap = *;
      WcorrMap = { 100.0799, 102.1580, 103.8778, 105.2502, 106.2882, 107.0060,
                   107.4188, 107.5439, 107.5439, 107.5439, 107.5439 }
    }
    NcorrMap= 0.800 {
      RlineMap = *;
      WcorrMap = { 116.8091, 118.8724, 120.5804, 121.9443, 122.9763, 123.6901,
                   124.1001, 124.2228, 124.2228, 124.2228, 124.2228 }
    }
    NcorrMap= 0.850 {
      RlineMap = *;
      WcorrMap = { 135.6708, 137.9187, 139.7752, 141.2518, 142.3627, 143.1228,
                   143.5481, 143.6591, 143.6591, 143.6591, 143.6591 }
    }
    NcorrMap= 0.900 {
      RlineMap = *;
      WcorrMap = { 157.0283, 159.6273, 161.7610, 163.4443, 164.6937, 165.5281,
                   165.9667, 166.0495, 166.0495, 166.0495, 166.0495 }
    }
    NcorrMap= 0.925 {
      RlineMap = *;
      WcorrMap = { 167.7171, 170.4948, 172.7683, 174.5537, 175.8698, 176.7370,
                   177.1761, 177.2440, 177.2440, 177.2440, 177.2440 }
    }
    NcorrMap= 0.950 {
      RlineMap = *;
      WcorrMap = { 180.5349, 182.6532, 184.4021, 185.7931, 186.8382, 187.5511,
                   187.9446, 188.0393, 188.0393, 188.0393, 188.0393 }
    }
    NcorrMap= 0.975 {
      RlineMap = *;
      WcorrMap = { 191.1595, 192.9175, 194.3772, 195.5477, 196.4376, 197.0574,
                   197.4157, 197.5237, 197.5237, 197.5237, 197.5237 }
    }
    NcorrMap= 1.000 {
      RlineMap = *;
      WcorrMap = { 202.6315, 203.5858, 204.3958, 205.0661, 205.5998, 206.0000,
                   206.2702, 206.4145, 206.4418, 206.4418, 206.4418 }
    }
    NcorrMap= 1.025 {
      RlineMap = *;
      WcorrMap = { 209.9986, 210.5917, 211.1029, 211.5321, 211.8825, 212.1554,
                   212.3516, 212.4735, 212.5220, 212.5227, 212.5227 }
    }
    NcorrMap= 1.050 {
      RlineMap = *;
      WcorrMap = { 216.6847, 217.0279, 217.3287, 217.5860, 217.8015, 217.9767,
                   218.1106, 218.2041, 218.2586, 218.2739, 218.2739 }
    }
  }
}

Table TB_eff(real alphaMap, real NcorrMap, real RlineMap) {
  alphaMap= 0.000 {
    NcorrMap= 0.500 {
      RlineMap = {     1.000,  1.200,  1.400,  1.600,  1.800,  2.000,  2.200,
                       2.400,  2.600,  2.800,  3.000 }
      effAdiabMap = { 0.6753, 0.6913, 0.7016, 0.7050, 0.7004, 0.6864, 0.6570,
                      0.6044, 0.5236, 0.4075, 0.2467 }
    }
    NcorrMap= 0.600 {
```

```
    RlineMap = *;
    effAdiabMap = { 0.6953, 0.7094, 0.7184, 0.7214, 0.7176, 0.7058, 0.6812,
                    0.6378, 0.5717, 0.4783, 0.3512 }
  }
  NcorrMap= 0.700 {
    RlineMap = *;
    effAdiabMap = { 0.7248, 0.7359, 0.7429, 0.7452, 0.7424, 0.7335, 0.7154,
                    0.6838, 0.6366, 0.5713, 0.4848 }
  }
  NcorrMap= 0.750 {
    RlineMap = *;
    effAdiabMap = { 0.7427, 0.7533, 0.7600, 0.7627, 0.7606, 0.7533, 0.7379,
                    0.7108, 0.6703, 0.6147, 0.5414 }
  }
  NcorrMap= 0.800 {
    RlineMap = *;
    effAdiabMap = { 0.7634, 0.7736, 0.7804, 0.7834, 0.7822, 0.7762, 0.7630,
                    0.7394, 0.7041, 0.6556, 0.5920 }
  }
  NcorrMap= 0.850 {
    RlineMap = *;
    effAdiabMap = { 0.7891, 0.8008, 0.8090, 0.8134, 0.8136, 0.8092, 0.7974,
                    0.7754, 0.7417, 0.6950, 0.6335 }
  }
  NcorrMap= 0.900 {
    RlineMap = *;
    effAdiabMap = { 0.8139, 0.8280, 0.8385, 0.8449, 0.8469, 0.8439, 0.8333,
                    0.8117, 0.7779, 0.7303, 0.6671 }
  }
  NcorrMap= 0.925 {
    RlineMap = *;
    effAdiabMap = { 0.8206, 0.8356, 0.8469, 0.8541, 0.8567, 0.8544, 0.8442,
                    0.8229, 0.7892, 0.7416, 0.6783 }
  }
  NcorrMap= 0.950 {
    RlineMap = *;
    effAdiabMap = { 0.8403, 0.8512, 0.8593, 0.8643, 0.8660, 0.8641, 0.8566,
                    0.8415, 0.8179, 0.7852, 0.7423 }
  }
  NcorrMap= 0.975 {
    RlineMap = *;
    effAdiabMap = { 0.8408, 0.8492, 0.8552, 0.8588, 0.8597, 0.8578, 0.8516,
                    0.8394, 0.8209, 0.7954, 0.7624 }
  }
  NcorrMap= 1.000 {
    RlineMap = *;
    effAdiabMap = { 0.8470, 0.8505, 0.8529, 0.8539, 0.8536, 0.8520, 0.8483,
                    0.8418, 0.8324, 0.8200, 0.8043 }
  }
  NcorrMap= 1.025 {
    RlineMap = *;
    effAdiabMap = { 0.8350, 0.8364, 0.8371, 0.8370, 0.8362, 0.8346, 0.8318,
                    0.8275, 0.8217, 0.8141, 0.8049 }
  }
  NcorrMap= 1.050 {
    RlineMap = *;
    effAdiabMap = { 0.8202, 0.8203, 0.8201, 0.8195, 0.8185, 0.8171, 0.8152,
                    0.8124, 0.8088, 0.8045, 0.7992 }
  }
}
alphaMap= 90.000 {
  NcorrMap= 0.500 {
    RlineMap = {     1.000,  1.200,  1.400,  1.600,  1.800,  2.000,  2.200,
```

```
                    2.400,  2.600,  2.800,  3.000 }
  effAdiabMap = { 0.6993, 0.7223, 0.7389, 0.7480, 0.7483, 0.7382, 0.7110,
                  0.6580, 0.5734, 0.4498, 0.2768 }
}
NcorrMap= 0.600 {
  RlineMap = *;
  effAdiabMap = { 0.7294, 0.7502, 0.7653, 0.7739, 0.7751, 0.7676, 0.7460,
                  0.7036, 0.6363, 0.5394, 0.4062 }
}
NcorrMap= 0.700 {
  RlineMap = *;
  effAdiabMap = { 0.7720, 0.7887, 0.8009, 0.8080, 0.8094, 0.8044, 0.7892,
                  0.7592, 0.7121, 0.6454, 0.5558 }
}
NcorrMap= 0.750 {
  RlineMap = *;
  effAdiabMap = { 0.7928, 0.8080, 0.8192, 0.8259, 0.8276, 0.8237, 0.8109,
                  0.7855, 0.7457, 0.6896, 0.6147 }
}
NcorrMap= 0.800 {
  RlineMap = *;
  effAdiabMap = { 0.8125, 0.8267, 0.8372, 0.8436, 0.8456, 0.8426, 0.8318,
                  0.8099, 0.7757, 0.7275, 0.6635 }
}
NcorrMap= 0.850 {
  RlineMap = *;
  effAdiabMap = { 0.8214, 0.8356, 0.8463, 0.8529, 0.8553, 0.8528, 0.8429,
                  0.8224, 0.7901, 0.7446, 0.6843 }
}
NcorrMap= 0.900 {
  RlineMap = *;
  effAdiabMap = { 0.8276, 0.8430, 0.8546, 0.8622, 0.8652, 0.8633, 0.8537,
                  0.8332, 0.8005, 0.7544, 0.6931 }
}
NcorrMap= 0.925 {
  RlineMap = *;
  effAdiabMap = { 0.8245, 0.8403, 0.8521, 0.8599, 0.8630, 0.8611, 0.8514,
                  0.8306, 0.7975, 0.7507, 0.6887 }
}
NcorrMap= 0.950 {
  RlineMap = *;
  effAdiabMap = { 0.8341, 0.8451, 0.8531, 0.8581, 0.8598, 0.8578, 0.8504,
                  0.8352, 0.8117, 0.7790, 0.7364 }
}
NcorrMap= 0.975 {
  RlineMap = *;
  effAdiabMap = { 0.8379, 0.8463, 0.8523, 0.8559, 0.8568, 0.8549, 0.8486,
                  0.8365, 0.8179, 0.7925, 0.7596 }
}
NcorrMap= 1.000 {
  RlineMap = *;
  effAdiabMap = { 0.8470, 0.8505, 0.8529, 0.8539, 0.8536, 0.8520, 0.8483,
                  0.8418, 0.8324, 0.8200, 0.8043 }
}
NcorrMap= 1.025 {
  RlineMap = *;
  effAdiabMap = { 0.8350, 0.8364, 0.8371, 0.8370, 0.8362, 0.8346, 0.8318,
                  0.8275, 0.8217, 0.8141, 0.8049 }
}
NcorrMap= 1.050 {
  RlineMap = *;
  effAdiabMap = { 0.8202, 0.8203, 0.8201, 0.8195, 0.8185, 0.8171, 0.8152,
                  0.8124, 0.8088, 0.8045, 0.7992 }
```

```
    }
  }
}

Table TB_PR(real alphaMap, real NcorrMap, real RlineMap) {
  alphaMap= 0.000 {
    NcorrMap= 0.500 {
      RlineMap = {    1.000,   1.200,   1.400,   1.600,   1.800,   2.000,
                      2.200,   2.400,   2.600,   2.800,   3.000 }
      PratioMap = {  2.4769,  2.4288,  2.3620,  2.2778,  2.1774,  2.0627,
                     1.9284,  1.7711,  1.5958,  1.4083,  1.2146 }
    }
    NcorrMap= 0.600 {
      RlineMap = *;
      PratioMap = {  3.4633,  3.3778,  3.2643,  3.1248,  2.9619,  2.7787,
                     2.5679,  2.3253,  2.0595,  1.7802,  1.4973 }
    }
    NcorrMap= 0.700 {
      RlineMap = *;
      PratioMap = {  5.0821,  4.9375,  4.7554,  4.5391,  4.2923,  4.0194,
                     3.7106,  3.3602,  2.9800,  2.5826,  2.1813 }
    }
    NcorrMap= 0.750 {
      RlineMap = *;
      PratioMap = {  6.3490,  6.1658,  5.9371,  5.6667,  5.3594,  5.0204,
                     4.6377,  4.2042,  3.7342,  3.2431,  2.7467 }
    }
    NcorrMap= 0.800 {
      RlineMap = *;
      PratioMap = {  8.0021,  7.7686,  7.4792,  7.1388,  6.7532,  6.3287,
                     5.8504,  5.3097,  4.7237,  4.1114,  3.4919 }
    }
    NcorrMap= 0.850 {
      RlineMap = *;
      PratioMap = { 10.4899, 10.1976,  9.8249,  9.3786,  8.8669,  8.2989,
                     7.6539,  6.9201,  6.1229,  5.2899,  4.4495 }
    }
    NcorrMap= 0.900 {
      RlineMap = *;
      PratioMap = { 14.4564, 14.0970, 13.6074, 12.9977, 12.2808, 11.4715,
                    10.5377,  9.4621,  8.2878,  7.0614,  5.8306 }
    }
    NcorrMap= 0.925 {
      RlineMap = *;
      PratioMap = { 17.4426, 17.0500, 16.4870, 15.7661, 14.9034, 13.9183,
                    12.7692, 11.4347,  9.9718,  8.4425,  6.9104 }
    }
    NcorrMap= 0.950 {
      RlineMap = *;
      PratioMap = { 20.7403, 20.2486, 19.6093, 18.8329, 17.9324, 16.9227,
                    15.7626, 14.4263, 12.9562, 11.3983,  9.8011 }
    }
    NcorrMap= 0.975 {
      RlineMap = *;
      PratioMap = { 23.8298, 23.2601, 22.5536, 21.7200, 20.7705, 19.7178,
                    18.5212, 17.1524, 15.6466, 14.0424, 12.3810 }
    }
    NcorrMap= 1.000 {
      RlineMap = *;
      PratioMap = { 26.6962, 26.0933, 25.4175, 24.6733, 23.8656, 22.9999,
                    22.0495, 20.9930, 19.8436, 18.6163, 17.3267 }
    }
    NcorrMap= 1.025 {
```

```
    RlineMap = *;
    PratioMap = { 27.6439, 27.0687, 26.4522, 25.7969, 25.1052, 24.3798,
                  23.6033, 22.7614, 21.8600, 20.9058, 19.9057 }
  }
  NcorrMap= 1.050 {
    RlineMap = *;
    PratioMap = { 28.4663, 27.9667, 27.4460, 26.9054, 26.3460, 25.7690,
                  25.1640, 24.5225, 23.8472, 23.1399, 22.4038 }
  }
}
alphaMap= 90.000 {
  NcorrMap= 0.500 {
    RlineMap = {    1.000,   1.200,   1.400,   1.600,   1.800,   2.000,
                    2.200,   2.400,   2.600,   2.800,   3.000 }
    PratioMap = {  5.5200,  5.4081,  5.2155,  4.9486,  4.6162,  4.2292,
                   3.7722,  3.2417,  2.6710,  2.0954,  1.5493 }
  }
  NcorrMap= 0.600 {
    RlineMap = *;
    PratioMap = {  7.5640,  7.3974,  7.1268,  6.7607,  6.3104,  5.7900,
                   5.1790,  4.4722,  3.7116,  2.9418,  2.2066 }
  }
  NcorrMap= 0.700 {
    RlineMap = *;
    PratioMap = { 10.2643, 10.0087,  9.6410,  9.1703,  8.6086,  7.9699,
                   7.2293,  6.3765,  5.4525,  4.5013,  3.5672 }
  }
  NcorrMap= 0.750 {
    RlineMap = *;
    PratioMap = { 12.0520, 11.7493, 11.3275, 10.7965, 10.1683,  9.4573,
                   8.6354,  7.6892,  6.6601,  5.5926,  4.5321 }
  }
  NcorrMap= 0.800 {
    RlineMap = *;
    PratioMap = { 14.2753, 13.9203, 13.4356, 12.8317, 12.1210, 11.3187,
                  10.3925,  9.3260,  8.1618,  6.9465,  5.7278 }
  }
  NcorrMap= 0.850 {
    RlineMap = *;
    PratioMap = { 16.9112, 16.5159, 15.9649, 15.2698, 14.4454, 13.5091,
                  12.4221, 11.1638,  9.7852,  8.3414,  6.8896 }
  }
  NcorrMap= 0.900 {
    RlineMap = *;
    PratioMap = { 20.7368, 20.3081, 19.6692, 18.8350, 17.8246, 16.6613,
                  15.2936, 13.6955, 11.9366, 10.0936,  8.2454 }
  }
  NcorrMap= 0.925 {
    RlineMap = *;
    PratioMap = { 22.5680, 22.1257, 21.4443, 20.5401, 19.4347, 18.1546,
                  16.6421, 14.8683, 12.9140, 10.8676,  8.8203 }
  }
  NcorrMap= 0.950 {
    RlineMap = *;
    PratioMap = { 24.1283, 23.5795, 22.8503, 21.9531, 20.9035, 19.7197,
                  18.3522, 16.7697, 15.0246, 13.1743, 11.2785 }
  }
  NcorrMap= 0.975 {
    RlineMap = *;
    PratioMap = { 25.7180, 25.1128, 24.3564, 23.4593, 22.4338, 21.2942,
                  19.9953, 18.5065, 16.8666, 15.1186, 13.3083 }
  }
  NcorrMap= 1.000 {
```

```
      RlineMap = *;
      PratioMap = { 26.6962, 26.0933, 25.4175, 24.6733, 23.8656, 22.9999,
                    22.0495, 20.9930, 19.8436, 18.6163, 17.3267 }
    }
    NcorrMap= 1.025 {
      RlineMap = *;
      PratioMap = { 27.6439, 27.0687, 26.4522, 25.7969, 25.1052, 24.3798,
                    23.6033, 22.7614, 21.8600, 20.9058, 19.9057 }
    }
    NcorrMap= 1.050 {
      RlineMap = *;
      PratioMap = { 28.4663, 27.9667, 27.4460, 26.9054, 26.3460, 25.7690,
                    25.1640, 24.5225, 23.8472, 23.1399, 22.4038 }
    }
  }
 }
}
}
```

```
//-------------------------------------------------------------------------
// File Name:     turbojet_HPT.map
// This data Map is generated from an existing NEPP map.
// The original NEPP map tables were:
//   1269    BASELINE HP TURBINE EFFICIENCY VS. PR, RPM, AND AREA
//   1270    BASELINE HP TURBINE FLOW FUNCTION VS. PR, RPM, AND AREA
//
//-------------------------------------------------------------------------

//  ---------------------------------------------------------------
//  |    Declaration of a new SubElement instance called S_map
//  |    that is of the type: S_map
//  ---------------------------------------------------------------


Subelement TurbineNeppMap S_map {

  s_parmMap  = 1.0;
  a_parmMap  = 0.0;
  s_parmGeom = 1.0;
  a_parmGeom = 0.0;
  s_RNI = 1.0;
  a_RNI = 0.0;
  effDes     = 0.9328;
  parmGeomDes= 1.0;
  parmMapDes = 5.000;
  parmNcDes  = 100.0;
  parmGeom = 1.0;
  parmMap = 5.000;


//  ---------------------------------------------------------------
//  |Declaration of a new SubElement instance called TurbineEfficiencyMap
//  |that is of the type: TurbineEfficiencyMap
//  ---------------------------------------------------------------

Subelement TurbineEfficiencyMap S_eff {

 Table TB_eff(real PgeomDesign, real NcDes, real PRdes) {

   PgeomDesign = 1.0 {
     NcDes=  60.000 {
       PRdes =   { 3.000,   3.250,   3.500,   3.750,   4.000,   4.250,   4.500,
                   4.750,   5.000,   5.250,   5.500,   5.750,   6.000,   6.250,
                   6.500,   6.750,   7.000,   7.250,   7.500,   8.000 }
       effMap = { 0.8460,  0.8405,  0.8349,  0.8296,  0.8249,  0.8206,  0.8165,
                  0.8127,  0.8092,  0.8060,  0.8030,  0.8002,  0.7976,  0.7953,
                  0.7931,  0.7911,  0.7892,  0.7875,  0.7858,  0.7826 }
     }
     NcDes=  70.000 {
       PRdes = *;
       effMap = { 0.8879,  0.8842,  0.8804,  0.8769,  0.8735,  0.8701,  0.8670,
                  0.8640,  0.8614,  0.8590,  0.8568,  0.8548,  0.8529,  0.8511,
                  0.8495,  0.8479,  0.8460,  0.8436,  0.8414,  0.8373 }
     }
     NcDes=  80.000 {
       PRdes = *;
       effMap = { 0.9125,  0.9111,  0.9096,  0.9078,  0.9055,  0.9034,  0.9014,
                  0.8995,  0.8979,  0.8964,  0.8950,  0.8936,  0.8924,  0.8903,
                  0.8877,  0.8853,  0.8830,  0.8808,  0.8787,  0.8749 }
     }
     NcDes=  90.000 {
       PRdes = *;
```

```
    effMap = { 0.9228,  0.9242,  0.9250,  0.9247,  0.9240,  0.9232,  0.9224,
               0.9217,  0.9210,  0.9203,  0.9197,  0.9188,  0.9162,  0.9137,
               0.9113,  0.9091,  0.9070,  0.9050,  0.9031,  0.8980 }
  }
  NcDes= 100.000 {
    PRdes = *;
    effMap = { 0.9215,  0.9258,  0.9289,  0.9304,  0.9313,  0.9319,  0.9323,
               0.9326,  0.9328,  0.9329,  0.9330,  0.9311,  0.9288,  0.9266,
               0.9245,  0.9225,  0.9206,  0.9188,  0.9161,  0.9107 }
  }
  NcDes= 110.000 {
    PRdes = *;
    effMap = { 0.9106,  0.9176,  0.9232,  0.9267,  0.9292,  0.9312,  0.9327,
               0.9340,  0.9351,  0.9361,  0.9369,  0.9349,  0.9329,  0.9311,
               0.9293,  0.9276,  0.9259,  0.9239,  0.9212,  0.9161 }
  }
}
PgeomDesign = 2.0 {
  NcDes=  60.000 {
    PRdes =   { 3.000,   3.250,   3.500,   3.750,   4.000,   4.250,   4.500,
                4.750,   5.000,   5.250,   5.500,   5.750,   6.000,   6.250,
                6.500,   6.750,   7.000,   7.250,   7.500,   8.000 }
    effMap = { 0.8460,  0.8405,  0.8349,  0.8296,  0.8249,  0.8206,  0.8165,
               0.8127,  0.8092,  0.8060,  0.8030,  0.8002,  0.7976,  0.7953,
               0.7931,  0.7911,  0.7892,  0.7875,  0.7858,  0.7826 }
  }
  NcDes=  70.000 {
    PRdes = *;
    effMap = { 0.8879,  0.8842,  0.8804,  0.8769,  0.8735,  0.8701,  0.8670,
               0.8640,  0.8614,  0.8590,  0.8568,  0.8548,  0.8529,  0.8511,
               0.8495,  0.8479,  0.8460,  0.8436,  0.8414,  0.8373 }
  }
  NcDes=  80.000 {
    PRdes = *;
    effMap = { 0.9125,  0.9111,  0.9096,  0.9078,  0.9055,  0.9034,  0.9014,
               0.8995,  0.8979,  0.8964,  0.8950,  0.8936,  0.8924,  0.8903,
               0.8877,  0.8853,  0.8830,  0.8808,  0.8787,  0.8749 }
  }
  NcDes=  90.000 {
    PRdes = *;
    effMap = { 0.9228,  0.9242,  0.9250,  0.9247,  0.9240,  0.9232,  0.9224,
               0.9217,  0.9210,  0.9203,  0.9197,  0.9188,  0.9162,  0.9137,
               0.9113,  0.9091,  0.9070,  0.9050,  0.9031,  0.8980 }
  }
  NcDes= 100.000 {
    PRdes = *;
    effMap = { 0.9215,  0.9258,  0.9289,  0.9304,  0.9313,  0.9319,  0.9323,
               0.9326,  0.9328,  0.9329,  0.9330,  0.9311,  0.9288,  0.9266,
               0.9245,  0.9225,  0.9206,  0.9188,  0.9161,  0.9107 }
  }
  NcDes= 110.000 {
    PRdes = *;
    effMap = { 0.9106,  0.9176,  0.9232,  0.9267,  0.9292,  0.9312,  0.9327,
               0.9340,  0.9351,  0.9361,  0.9369,  0.9349,  0.9329,  0.9311,
               0.9293,  0.9276,  0.9259,  0.9239,  0.9212,  0.9161 }
  }
}

PgeomDesign.interp = "linear" ;
PgeomDesign.extrap = "none" ;

NcDes.interp = "linear" ;
NcDes.extrap = "none" ;
```

```
  PRdes.interp = "linear" ;
  PRdes.extrap = "none" ;
}


Table TB_Wp(real PgeomDesign, real NcDes, real PRdes) {

  PgeomDesign = 1.0 {
    NcDes=  60.000 {
      PRdes =   { 3.000,   3.250,   3.500,   3.750,   4.000,   4.250,   4.500,
                  4.750,   5.000,   5.250,   5.500,   5.750,   6.000,   6.250,
                  6.500,   6.750,   7.000,   7.250,   7.500,   8.000 }
      WcMap = {  30.446,  30.533,  30.568,  30.572,  30.572,  30.572,  30.572,
                 30.572,  30.572,  30.572,  30.572,  30.572,  30.572,  30.572,
                 30.572,  30.572,  30.572,  30.572,  30.572,  30.572 }
    }
    NcDes=  70.000 {
      PRdes = *;
      WcMap = {  30.299,  30.413,  30.480,  30.516,  30.529,  30.530,  30.530,
                 30.530,  30.530,  30.530,  30.530,  30.530,  30.530,  30.530,
                 30.530,  30.530,  30.530,  30.530,  30.530,  30.530 }
    }
    NcDes=  80.000 {
      PRdes = *;
      WcMap = {  30.120,  30.239,  30.317,  30.368,  30.398,  30.415,  30.421,
                 30.421,  30.421,  30.421,  30.421,  30.421,  30.421,  30.421,
                 30.421,  30.421,  30.421,  30.421,  30.421,  30.421 }
    }
    NcDes=  90.000 {
      PRdes = *;
      WcMap = {  30.014,  30.124,  30.201,  30.253,  30.288,  30.311,  30.325,
                 30.333,  30.337,  30.337,  30.337,  30.337,  30.337,  30.337,
                 30.337,  30.337,  30.337,  30.337,  30.337,  30.337 }
    }
    NcDes= 100.000 {
      PRdes = *;
      WcMap = {  29.856,  29.948,  30.013,  30.059,  30.091,  30.113,  30.128,
                 30.139,  30.145,  30.149,  30.150,  30.150,  30.150,  30.150,
                 30.150,  30.150,  30.150,  30.150,  30.150,  30.150 }
    }
    NcDes= 110.000 {
      PRdes = *;
      WcMap = {  29.799,  29.870,  29.920,  29.955,  29.979,  29.997,  30.009,
                 30.017,  30.023,  30.026,  30.028,  30.029,  30.029,  30.029,
                 30.029,  30.029,  30.029,  30.029,  30.029,  30.029 }
    }
  }
  PgeomDesign = 2.0 {
    NcDes=  60.000 {
      PRdes =   { 3.000,   3.250,   3.500,   3.750,   4.000,   4.250,   4.500,
                  4.750,   5.000,   5.250,   5.500,   5.750,   6.000,   6.250,
                  6.500,   6.750,   7.000,   7.250,   7.500,   8.000 }
      WcMap = {  30.446,  30.533,  30.568,  30.572,  30.572,  30.572,  30.572,
                 30.572,  30.572,  30.572,  30.572,  30.572,  30.572,  30.572,
                 30.572,  30.572,  30.572,  30.572,  30.572,  30.572 }
    }
    NcDes=  70.000 {
      PRdes = *;
      WcMap = {  30.299,  30.413,  30.480,  30.516,  30.529,  30.530,  30.530,
                 30.530,  30.530,  30.530,  30.530,  30.530,  30.530,  30.530,
                 30.530,  30.530,  30.530,  30.530,  30.530,  30.530 }
    }
```

```
      NcDes=  80.000 {
        PRdes = *;
        WcMap = {  30.120,  30.239,  30.317,  30.368,  30.398,  30.415,  30.421,
                   30.421,  30.421,  30.421,  30.421,  30.421,  30.421,  30.421,
                   30.421,  30.421,  30.421,  30.421,  30.421,  30.421 }
      }
      NcDes=  90.000 {
        PRdes = *;
        WcMap = {  30.014,  30.124,  30.201,  30.253,  30.288,  30.311,  30.325,
                   30.333,  30.337,  30.337,  30.337,  30.337,  30.337,  30.337,
                   30.337,  30.337,  30.337,  30.337,  30.337,  30.337 }
      }
      NcDes= 100.000 {
        PRdes = *;
        WcMap = {  29.856,  29.948,  30.013,  30.059,  30.091,  30.113,  30.128,
                   30.139,  30.145,  30.149,  30.150,  30.150,  30.150,  30.150,
                   30.150,  30.150,  30.150,  30.150,  30.150,  30.150 }
      }
      NcDes= 110.000 {
        PRdes = *;
        WcMap = {  29.799,  29.870,  29.920,  29.955,  29.979,  29.997,  30.009,
                   30.017,  30.023,  30.026,  30.028,  30.029,  30.029,  30.029,
                   30.029,  30.029,  30.029,  30.029,  30.029,  30.029 }
      }
    }

    PgeomDesign.interp = "linear" ;
    PgeomDesign.extrap = "none" ;

    NcDes.interp = "linear" ;
    NcDes.extrap = "none" ;

    PRdes.interp = "linear" ;
    PRdes.extrap = "none" ;
  }
}
}
```

```
// File Name:     turbojet.view_page
// <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
//          General Output DataViewer Specification
// <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

OutFileStream pageStream { filename = "turbojet.output"; }

DataViewer PageViewer page {

GroupBlock TnBGroup {

   SimpleBlock title {
body =   "\f*****************************************************************************************************************************************\n" +
         "Date:????????    Time:????????    Model:??????????????????????????????????????????????????????????????????????    converge = ?   CASE:????\n" +
         "Version:??????????????????    Gas Package:??????    iter/pass/Jacb/Broy=???/???/??/??    Run by:??????????????????                ";
vars = { "date", "timeOfDay", "MODELNAME", "solver.converged", "CASE", "VERSION", "THERMPACKAGE",
          "solver.iterationCounter", "solver.passCounter", "solver.numJacobians", "solver.numBroydens", "USER" }
   }

   SimpleBlock summary {
body = "                                        SUMMARY OUTPUT DATA \n" +
       "    MN       alt    dTamb         W         Fn      TSFC      Wfuel       OPR       T3       T4\n" +
       " ?.???   ?????.?   ???.??   ????.??   ??????.?   ??.????   ?????.??    ??.???   ????.?   ????.?";
vars = { "PERF.MN", "PERF.alt", "PERF.dTs", "PERF.W", "PERF.Fn", "PERF.TSFC", "PERF.WfuelHour",
         "PERF.OPR", "FS_3.Tt", "FS_4.Tt" }
   }

   anchor = "title";
   title.bottom = "summary";
   addGutter = TRUE;
} // end TnBGroup

LinkColTBlock inputs {
   titleBody = "                                          FLOW STATION DATA";
   titleVars = { }
   compType = "FluidOutputPort";
   compTypeFormat = "??????????????????????";
   excludeCompTypes = { "InterStageBleedOutPort", "BleedOutPort" }     //remove to include bleed ports
   compAttr = { "W:????.??", "Pt:????.???", "Tt:?????.??", "ht:????.??", "FAR:?.????",
                "Wc:????.??", "Ps:????.???", "Ts:?????.??", "Aphy:?????.?", "MN:?.????", "gamt:??.?????", "getConstituencyTotal(\"H2O\"):??.?????=H2O" }
}


GroupBlock Turbomachinery {
   DColTBlock comps {
      titleBody = "TURBOMACHINERY PERFORMANCE DATA";
      compType = "Compressor";
      compTypeFormat = "????????";
      compAttr = { "Fl_I.Wc:????.??=Wc", "PR:??.???", "eff:??.????", "Nc:?????.???",
                   "TR:??.????", "effPoly:??.????=efPoly", "pwr:???????.?", "SMN:???.??", "SMW:???.??" }
   }
   DColTBlock turbs {
      compType = "Turbine";
      compTypeFormat = "????????";
      compAttr = { "Fl_I.Wc:????.??=Wc", "PR:??.???", "eff:??.????", "Np:?????.???",
                   "TR:??.????", "effPoly:??.????=efPoly", "pwr:???????.?" }
      showColHeader=FALSE;
   }
   anchor = "comps";
   comps.bottom = "turbs";
   addGutter = FALSE;
}  //end of Turbomachinery block

GroupBlock MapsGroup {
   DColTBlock compMaps {
      titleBody = "TURBOMACHINERY MAP DATA";
      compType = "Compressor";
      compTypeFormat = "????????";
      compAttr = { "S_map.WcMap:????.??=WcMap", "S_map.PRmap:??.???=PRmap", "S_map.effMap:??.????=effMap",
                   "S_map.NcMap:?????.???=NcMap", "S_map.RlineMap:???.????=R/Parm", "S_map.s_WcDes:?????.????=s_WcDes",
                   "S_map.s_PRdes:???.????=s_PRdes", "S_map.s_effDes:?.????=s_effDes", "S_map.s_NcDes:???.????=s_NcDes" }
   }
   DColTBlock turbMaps {
      compType = "Turbine";
      compTypeFormat = "????????";
      compAttr = { "S_map.WpMap:????.??=WpMap", "S_map.PRmap:??.???=PRmap", "S_map.effMap:??.????=effMap",
                   "S_map.NpMap:?????.???=NpMap", "S_map.PRmap:???.????=parmMap", "S_map.s_Wp:?????.????=s_WpDes",
                   "S_map.s_dPqP:???.????=s_PRdes", "S_map.s_eff:?.????=s_effDes", "S_map.s_Np:???.????=s_NpDes" }
      showColHeader=FALSE;
   }
   anchor = "compMaps";
   compMaps.bottom = "turbMaps";
   addGutter = FALSE;
}  //end of MapsGroup
```


NASA/TM—2007-214690

```
GroupBlock ABblock {  //=============================================================================================

   GroupBlock Components {   //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      DColTBlock inlets {
         titleBody = "===INLETS====    eRam       Afs      Fram";
         compType = "Inlet";
         compTypeFormat = "????????????";
         compAttr = { "eRam:??.????", "Afs:?????.??", "Fram:??????.?" }
         showColHeader = FALSE;
      }
      DColTBlock ducts {
         titleBody = "====DUCTS====  dPnorm        MN      Aphy";
         compType = "Duct";
         compTypeFormat = "????????????";
         compAttr = { "dPqP:??.????", "Fl_I.MN:???.????=MNin", "Fl_I.Aphy:?????.??=Aphy" }
         showColHeader = FALSE;
      }
      DColTBlock splitters {
         titleBody = "==SPLITTERS==     BPR    dP/P 1    dP/P 2";
         compType = "Splitter";
         compTypeFormat = "????????????";
         compAttr = { "BPR:??.????", "dPqP1:???.????=dPpri/P", "dPqP2:???.????=dPsec/P" }
         showColHeader = FALSE;
      }
      DColTBlock shafts {
         titleBody = "===SHAFTS====   Nmech    trq in    pwr in";
         compType = "Shaft";
         compTypeFormat = "????????????";
         compAttr = { "Nmech:?????.?", "trqIn:??????.?", "pwrIn:??????.?" }
         showColHeader = FALSE;
      }
      DColTBlock loads {
         titleBody = "====LOADS====   Nload    Nratio   trqLoad      pwr";
         compType = "Load";
         compTypeFormat = "????????????";
         compAttr = { "Nload:?????.?", "NR:??.?????", "trqLoad:??????.?", "pwr:??????.?" }
         showColHeader = FALSE;
      }
      anchor = "inlets";
      inlets.bottom = "ducts";
      ducts.bottom = "splitters";
      splitters.bottom = "shafts";
      shafts.bottom = "loads";
   }  //end of Components

   GroupBlock BleedGroup {  //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      LinkColTBlock interStageBleeds {
         titleBody = "\nBLEEDS - interstg  Wb/Win    BldWk     BldP          W        Tt       ht        Pt";
         compType = "InterStageBleedOutPort";
         compTypeFormat = "??????????????????";
         compAttr = { "fracBldw:??.????=Wb/Win", "fracBldWork:??.????=dhb/dh", "fracBldP:??.????=dPb/dP",
                      "W:????.????", "Tt:?????.??", "ht:????.??", "Pt:????.???" }
         showColHeader = FALSE;
      }
      LinkColTBlock outletBleeds {
         titleBody = "\nBLEEDS - output    Wb/Win   hscale   Pscale          W        Tt       ht        Pt";
         compType = "BleedOutPort";
         compTypeFormat = "??????????????????";
         compAttr = { "fracW:??.????=Wb/Win", "hscale:??.????", "Pscale:??.????",
                      "W:????.????", "Tt:?????.??", "ht:????.??", "Pt:????.???" }
         showColHeader=FALSE;
      }
      interStageBleeds.bottom = "outletBleeds";
      addGutter = FALSE;
   }  //end of BleedGroup

   EmptyTextBlock space {  //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      width = 4;
      height = 0;
   }
   anchor = "Components";
   Components.right = "space";
   space.right = "BleedGroup";
}  //end of ABblock

GroupBlock Cblock {   //=============================================================================================

   DColTBlock burners {
      titleBody = "===BURNERS===   TtOut       eff    dPnorm      Wfuel       FAR";
      compType = "Burner";
      compTypeFormat = "????????????";
      compAttr = { "TtCombOut:????.??=TtOut", "eff:???.????", "dPqP:???.????=dPqP",
                   "Wfuel:???.?????", "FAR:??.?????" }
      showColHeader = FALSE;
   }
   DColTBlock mixers {
```

```
      titleBody = "===MIXERS====    Aout      MN1     MN2   MNout        Ps 1        Ps 2     imp mix     imp out";
      compType = "Mixer";
      compTypeFormat = "????????????";
      compAttr = {"Fl_O.Aphy:?????.??","Fl_I1.MN:??.???=MN_I1","Fl_I2.MN:??.???=MN_I2","Fl_O.MN:??.???=MN_out","Fl_I1.Ps","Fl_I2.Ps","impMixed","impOut"}
      showColHeader = FALSE;
   }
   DColTBlock nozzles {
      titleBody = "===NOZZLES===      PR       Cfg      CdTh       Cv        Ath      MNth     Vact        Fg";
      compType = "Nozzle";
      compTypeFormat = "????????????";
      compAttr = { "PR:???.???", "Cfg:???.????", "CdTh:???.????", "Cv:??.????",
                   "Ath:??????.??", "Fl_Th.MN:????.???=MNth", "Vactual:?????.?=Vact", "Fg:??????.?" }
      showColHeader = FALSE;
   }
   anchor = "burners";
   burners.bottom = "mixers";
   mixers.bottom  = "nozzles";
}  //end of Cblock

anchor = "TnBGroup";
TnBGroup.bottom = "inputs";
inputs.bottom = "Turbomachinery";
Turbomachinery.bottom = "MapsGroup";
MapsGroup.bottom = "ABblock";
ABblock.bottom = "Cblock";

pageWidth = 0;
pageHeight = 0;
outStreamHandle = "pageStream";

}
```

```
*******************************************************************************************************************************
Date:08/08/06    Time:10:56:55    Model:                                                          converge = 1   CASE:   0
Version:NPSS_1.6.4 - Rev: --    Gas Package: Janaf    iter/pass/Jacb/Broy=  3/  4/ 1/ 1    Run by:            sjones

                                    SUMMARY OUTPUT DATA
    MN       alt    dTamb         W         Fn      TSFC      Wfuel       OPR       T3       T4
 0.000       0.0     0.00    100.00    10654.8    1.0136   10800.00    19.600   1315.8   3065.0

                                    FLOW STATION DATA
                                  W        Pt        Tt        ht     FAR        Wc        Ps        Ts     Aphy      MN      gamt       H2O
FS_1 Ambient.Fl_O            100.00    14.696    518.67    -6.18  0.0000    100.00    14.696    518.67  -----.-  0.0000   1.40023   0.00000
FS_2 Inlet.Fl_O              100.00    14.402    518.67    -6.18  0.0000    102.04     0.000      0.00      0.0  0.0000   1.40023   0.00000
FS_3 HPC.Fl_O                100.00   288.042   1315.79   190.97  0.0000      8.13     0.000      0.00      0.0  0.0000   1.36092   0.00000
FS_4 Burner.Fl_O             103.00   273.640   3065.01   154.38  0.0300     13.45     0.000      0.00      0.0  0.0000   1.27466   0.03587
FS_5 HPT.Fl_O                103.00    89.596   2449.00   -37.02  0.0300     36.71     0.000      0.00      0.0  0.0000   1.29209   0.03591
FS_7 Duct1.Fl_O              103.00    88.700   2449.02   -37.02  0.0300     37.08     0.000      0.00      0.0  0.0000   1.29208   0.03591
FS_9 Nozzle.Fl_O             103.00    88.700   2449.06   -37.02  0.0300     37.08    48.460   2131.18    110.7  1.0000   1.29208   0.03591

TURBOMACHINERY PERFORMANCE DATA
              Wc      PR      eff        Nc       TR   efPoly       pwr     SMN     SMW
HPC       102.04  20.000   0.8500  10000.000   2.5368   0.8973  -27893.7   18.00   17.95
HPT        13.45   3.054   0.9200    180.627   1.2516   0.9004   27893.7

TURBOMACHINERY MAP DATA
            WcMap   PRmap   effMap     NcMap    R/Parm     s_WcDes   s_PRdes  s_effDes   s_NcDes
HPC        206.00  23.000   0.8520     1.000    2.0000      0.4953    0.8636    0.9977    0.0001
HPT         29.86   3.000   0.9215   100.000    3.0000      0.4504    0.9736    0.9984    0.0243

===INLETS====     eRam       Afs      Fram
Inlet           0.9800  -----.--       0.0

====DUCTS====  dPnorm        MN      Aphy
Duct1           0.0100    0.0000      0.00


===SHAFTS====    Nmech    trq in    pwr in
HP_Shaft      10000.0   14650.1   27893.7


===BURNERS===    TtOut       eff    dPnorm      Wfuel       FAR
Burner         3065.01    1.0000    0.0500    3.00000   0.03000


===NOZZLES===       PR       Cfg      CdTh       Cv        Ath      MNth     Vact        Fg
Nozzle           6.036    0.9472    1.0000   0.9900     110.70     1.000   2160.7   10654.8
```

```
*******************************************************************************************************************************
Date:08/08/06    Time:10:56:55    Model:                                                          converge = 1   CASE:   0
Version:NPSS_1.6.4 - Rev: --    Gas Package: Janaf    iter/pass/Jacb/Broy=  1/  1/ 0/ 0    Run by:            sjones

                                    SUMMARY OUTPUT DATA
    MN       alt    dTamb         W         Fn      TSFC      Wfuel       OPR       T3       T4
 0.000       0.0     0.00    100.00    10654.8    1.0136   10800.00    19.600   1315.8   3065.0

                                    FLOW STATION DATA
                                  W        Pt        Tt        ht     FAR        Wc        Ps        Ts     Aphy      MN      gamt       H2O
FS_1 Ambient.Fl_O            100.00    14.696    518.67    -6.18  0.0000    100.00    14.696    518.67  -----.-  0.0000   1.40023   0.00000
FS_2 Inlet.Fl_O              100.00    14.402    518.67    -6.18  0.0000    102.04     0.000      0.00      0.0  0.0000   1.40023   0.00000
FS_3 HPC.Fl_O                100.00   288.042   1315.79   190.97  0.0000      8.13     0.000      0.00      0.0  0.0000   1.36092   0.00000
FS_4 Burner.Fl_O             103.00   273.640   3065.01   154.38  0.0300     13.45     0.000      0.00      0.0  0.0000   1.27466   0.03587
FS_5 HPT.Fl_O                103.00    89.596   2449.00   -37.02  0.0300     36.71     0.000      0.00      0.0  0.0000   1.29209   0.03591
FS_7 Duct1.Fl_O              103.00    88.700   2449.02   -37.02  0.0300     37.08     0.000      0.00      0.0  0.0000   1.29208   0.03591
FS_9 Nozzle.Fl_O             103.00    88.700   2449.06   -37.02  0.0300     37.08    48.460   2131.18    110.7  1.0000   1.29208   0.03591

TURBOMACHINERY PERFORMANCE DATA
              Wc      PR      eff        Nc       TR   efPoly       pwr     SMN     SMW
HPC       102.04  20.000   0.8500  10000.000   2.5368   0.8973  -27893.7   18.00   17.95
HPT        13.45   3.054   0.9200    180.627   1.2516   0.9004   27893.7
```

TURBOMACHINERY MAP DATA

	WcMap	PRmap	effMap	NcMap	R/Parm	s_WcDes	s_PRdes	s_effDes	s_NcDes
HPC	206.00	23.000	0.8520	1.000	2.0000	0.4953	0.8636	0.9977	0.0001
HPT	29.86	3.000	0.9215	100.000	3.0000	0.4504	0.9736	0.9984	0.0243

===INLETS====	eRam	Afs	Fram
Inlet	0.9800	-----.--	0.0

====DUCTS====	dPnorm	MN	Aphy
Duct1	0.0100	0.0000	0.00

===SHAFTS====	Nmech	trq in	pwr in
HP_Shaft	10000.0	14650.1	27893.7

===BURNERS===	TtOut	eff	dPnorm	Wfuel	FAR
Burner	3065.01	1.0000	0.0500	3.00000	0.03000

===NOZZLES===	PR	Cfg	CdTh	Cv	Ath	MNth	Vact	Fg
Nozzle	6.036	0.9472	1.0000	0.9900	110.70	1.000	2160.7	10654.8

****

Date:08/08/06 Time:10:56:55 Model: converge = 1 CASE: 0

Version:NPSS_1.6.4 - Rev: -- Gas Package: Janaf iter/pass/Jacb/Broy= 9/ 13/ 1/ 7 Run by: sjones

SUMMARY OUTPUT DATA

MN	alt	dTamb	W	Fn	TSFC	Wfuel	OPR	T3	T4
1.000	20000.0	0.00	85.53	7209.4	1.3410	9668.20	19.614	1359.2	3168.2

FLOW STATION DATA

	W	Pt	Tt	ht	FAR	Wc	Ps	Ts	Aphy	MN	gamt	H2O
FS_1 Ambient.Fl_O	85.53	12.788	536.96	-1.79	0.0000	100.01	6.753	447.35	291.4	1.0000	1.39997	0.00000
FS_2 Inlet.Fl_O	85.53	12.532	536.96	-1.79	0.0000	102.05	0.000	0.00	0.0	0.0000	1.39997	0.00000
FS_3 HPC.Fl_O	85.53	250.816	1359.19	202.22	0.0000	8.11	0.000	0.00	0.0	0.0000	1.35832	0.00000
FS_4 Burner.Fl_O	88.21	238.276	3168.19	163.49	0.0314	13.45	0.000	0.00	0.0	0.0000	1.27102	0.03747
FS_5 HPT.Fl_O	88.21	78.150	2537.86	-34.31	0.0314	36.69	0.000	0.00	0.0	0.0000	1.28858	0.03753
FS_7 Duct1.Fl_O	88.21	77.368	2537.88	-34.31	0.0314	37.07	0.000	0.00	0.0	0.0000	1.28858	0.03753
FS_9 Nozzle.Fl_O	88.21	77.368	2537.92	-34.31	0.0314	37.07	42.342	2211.97	110.7	1.0000	1.28858	0.03753

TURBOMACHINERY PERFORMANCE DATA

	Wc	PR	eff	Nc	TR	efPoly	pwr	SMN	SMW
HPC	102.05	20.014	0.8499	10001.144	2.5313	0.8971	-24687.5	17.93	17.88
HPT	13.45	3.049	0.9199	180.787	1.2484	0.8988	24687.5		

TURBOMACHINERY MAP DATA

	WcMap	PRmap	effMap	NcMap	R/Parm	s_WcDes	s_PRdes	s_effDes	s_NcDes
HPC	206.02	23.017	0.8519	1.000	1.9976	0.4953	0.8636	0.9977	0.0001
HPT	29.86	2.995	0.9214	100.089	2.9949	0.4504	0.9736	0.9984	0.0243

===INLETS====	eRam	Afs	Fram
Inlet	0.9800	291.43	2757.1

====DUCTS====	dPnorm	MN	Aphy
Duct1	0.0100	0.0000	0.00

===SHAFTS====	Nmech	trq in	pwr in
HP_Shaft	10175.9	12742.0	24687.5

===BURNERS===	TtOut	eff	dPnorm	Wfuel	FAR
Burner	3168.19	1.0000	0.0500	2.68561	0.03140

===NOZZLES===	PR	Cfg	CdTh	Cv	Ath	MNth	Vact	Fg
Nozzle	11.456	0.9029	1.0000	0.9900	110.70	1.000	2198.2	9966.6

```
□*****************************************************************************************************************************************************
Date:08/08/06    Time:10:56:55    Model:                                                                converge = 1   CASE:   0
Version:NPSS_1.6.4 - Rev: --    Gas Package: Janaf    iter/pass/Jacb/Broy=  2/  7/ 1/ 0    Run by:            sjones

                                    SUMMARY OUTPUT DATA
    MN       alt    dTamb         W         Fn      TSFC      Wfuel       OPR       T3       T4
 1.000   20000.0     0.00     86.09     7327.4    1.3500    9892.09    19.852   1367.4   3200.0

                                    FLOW STATION DATA
                                 W        Pt        Tt        ht     FAR       Wc        Ps        Ts     Aphy      MN      gamt       H2O
FS_1 Ambient.Fl_O            86.09    12.788    536.96     -1.79  0.0000   100.67     6.753    447.35    293.3  1.0000   1.39997   0.00000
FS_2 Inlet.Fl_O              86.09    12.532    536.96     -1.79  0.0000   102.72     0.000      0.00      0.0  0.0000   1.39997   0.00000
FS_3 HPC.Fl_O                86.09   253.856   1367.43    204.36  0.0000     8.09     0.000      0.00      0.0  0.0000   1.35784   0.00000
FS_4 Burner.Fl_O             88.84   241.163   3199.97    164.91  0.0319    13.45     0.000      0.00      0.0  0.0000   1.26988   0.03806
FS_5 HPT.Fl_O                88.84    79.139   2565.37    -34.85  0.0319    36.69     0.000      0.00      0.0  0.0000   1.28747   0.03813
FS_7 Duct1.Fl_O              88.84    78.348   2565.39    -34.85  0.0319    37.06     0.000      0.00      0.0  0.0000   1.28747   0.03813
FS_9 Nozzle.Fl_O             88.84    78.348   2565.43    -34.85  0.0319    37.06    42.902   2237.03    110.7  1.0000   1.28747   0.03813

TURBOMACHINERY PERFORMANCE DATA
               Wc      PR      eff        Nc       TR   efPoly        pwr     SMN     SMW
HPC        102.72  20.257   0.8461  10055.712   2.5466   0.8946   -25110.3   17.25   17.21
HPT         13.45   3.047   0.9199    180.869   1.2474   0.8982    25108.7

TURBOMACHINERY MAP DATA
            WcMap   PRmap   effMap      NcMap    R/Parm     s_WcDes   s_PRdes  s_effDes   s_NcDes
HPC        207.37  23.297   0.8481      1.006    2.0022      0.4953    0.8636    0.9977    0.0001
HPT         29.86   2.993   0.9214    100.134    2.9933      0.4504    0.9736    0.9984    0.0243

===INLETS====    eRam       Afs      Fram
Inlet          0.9800    293.35    2775.3

====DUCTS====  dPnorm        MN      Aphy
Duct1          0.0100    0.0000      0.00


===SHAFTS====   Nmech    trq in    pwr in
HP_Shaft      10231.4   12889.1   25108.7


===BURNERS===   TtOut       eff    dPnorm      Wfuel       FAR
Burner        3199.97    1.0000    0.0500    2.74780   0.03192


===NOZZLES===       PR       Cfg      CdTh       Cv        Ath      MNth     Vact        Fg
Nozzle          11.601    0.9019    1.0000   0.9900     110.70     1.000   2209.6   10102.7
```

Appendix C
Turbomachinery Maps

In the performance analysis of gas turbine engines, it is very desirable to have relationships that describe the variation in mass flow rate, rpm, pressure ratio, and temperature ratio for each turbomachine in the model. These characteristics are often calculated by complex codes using velocity diagrams, fluid mechanics, empirical correlations, and even computational fluid dynamics. The physics of turbomachinery flow simplify when corrected (or equivalent) airflow is used rather than absolute flow and corrected (or equivalent) speed is used for absolute rpm. The calculation of the relationship between corrected flow, corrected speed, pressure ratio, and temperature ratio (or efficiency) for a turbomachine results in a collection of data referred to as a "map".

Figure 2 shows a compressor map, normally plotted as pressure ratio (across the machine) versus corrected airflow into the machine. For multiple values of corrected rpm the relationship between pressure ratio and flow is shown and the line that connects the upper left of each speed characteristic is called the surge line. Each speed characteristic also has a lower limit: the line becomes vertical as the machine chokes and no further mass flow can be passed. Finally, values of constant adiabatic efficiency are plotted; these contours resemble, roughly speaking, a series of concentric ovals and are often referred to as "efficiency islands".

Figure 3 shows the same map with some of the data values identified. The application of such a specific compressor map to a complete engine system is quite limited; however, if the map could be scaled its usefulness would be dramatically increased. NPSS allows the map to be scaled up or down in pressure ratio, efficiency, flow, or any combination of the three. Assume that the design point is identified as 100 percent speed and 25 percent surge margin, resulting in an unscaled pressure ratio of 2.0, corrected airflow of 500 lb_m/s, and adiabatic efficiency of 87 percent. Further assume that the desired values of pressure ratio, flow, and efficiency at that point are 1.80, 475 lb_m/s, and 87.5 percent respectively. The ratio of the desired value to the unscaled value of each parameter determines the scale factor applied to the map (one scale factor for each parameter).

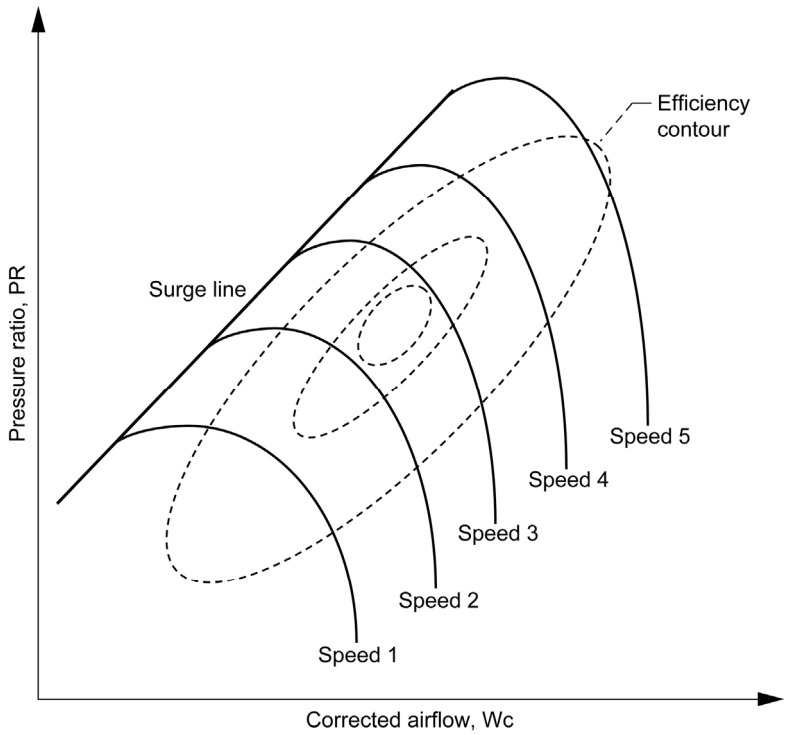

Figure 2.—Compressor map characteristics.

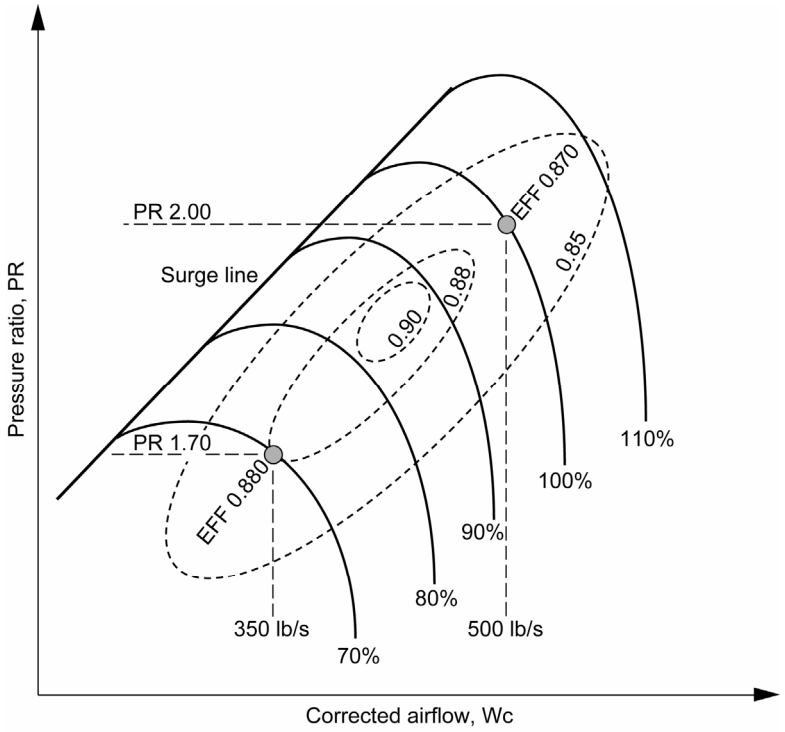

Figure 3.—Unscaled compressor map.

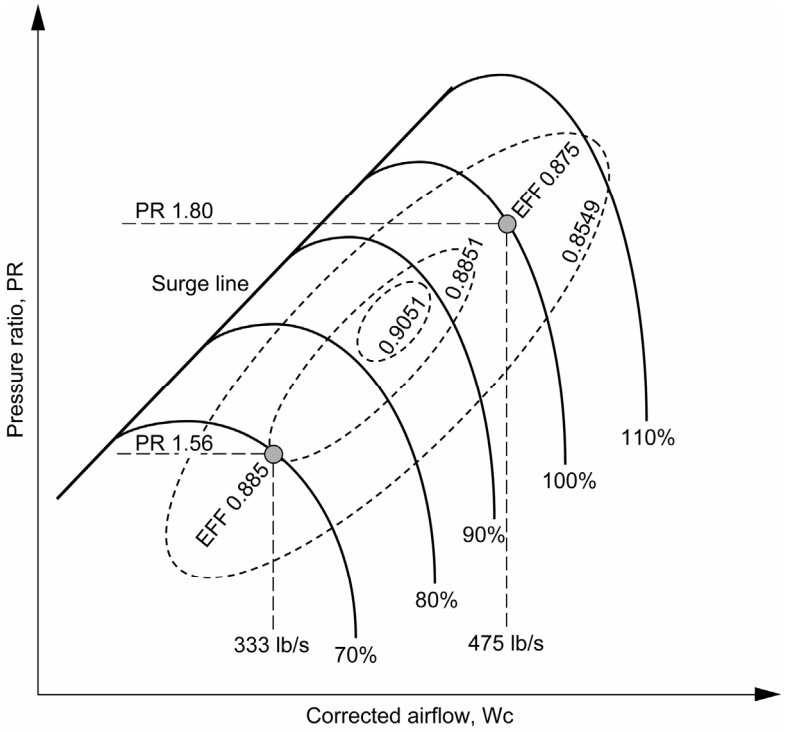

Figure 4.—Scaled compressor map.

Figure 4 is the scaled compressor map. The scale factors are calculated during the design point analysis and used during the performance analysis from the following equations:

$$\text{flow scale factor} = \frac{Wc_{\text{desired}}}{Wc_{\text{unscaled}}} = \frac{475\ \text{lb/s}}{500\ \text{lb/s}} = 0.950$$

$$\text{PR scale factor} = \frac{PR_{\text{desired}} - 1}{PR_{\text{unscaled}} - 1} = \frac{1.80 - 1}{2.00 - 1} = 0.800$$

$$\text{efficiency scale factor} = \frac{\eta_{\text{desired}}}{\eta_{\text{unscaled}}} = \frac{0.875}{0.870} = 1.006$$

Note that the limit for pressure ratio is 1, not 0, because the compressor/turbine requires/performs no work at that pressure ratio; this is why 1 is subtracted from both values in the pressure ratio scale factor equation. These scale factors are applied to every point in the map: thus all values of corrected flow are multiplied by 0.95, all efficiencies are increased by 0.6 percent, and pressure ratio minus one is reduced 20 percent as shown by the two points in figures 3 and 4.

There are two peculiarities of the compressor corrected speed characteristic because of its shape (resembling a candy cane): first, it is possible to have a particular value of pressure ratio and corrected speed occur at two values of corrected flow; second, pressure ratio is indeterminate when a particular value of corrected speed and flow occur on the vertical part of the speed characteristic. Because of this, compressor maps require an second independent parameter (the first is corrected speed) to uniquely determine the flow, PR and efficiency: for one commonly used format of compressor map in NPSS that parameter is called the R-line.

Figure 5 shows the compressor map with R-lines included. Typically the first R-line is given a value of 1.0 and coincides with the surge line. The remaining R-lines are roughly parallel to the surge line with increasing R-line value corresponding to higher surge margin. These R-lines have no physical meaning; they are simply mathematical constructs orthogonal to corrected speed so that any value of R-line and corrected speed will uniquely position the compressor operating point. In other words, given R-line value and corrected speed, the compressor pressure ratio, corrected flow, and adiabatic efficiency are known, and further, the derivatives of these parameters can be calculated with respect to R-line and corrected speed which is critical for computational convergence.

With regard to efficiency and corrected flow, the scaling of a turbine map (shown in fig. 6) is the same as with compressor maps. Since pressure ratio and speed uniquely determine corrected flow and efficiency, no R-lines are required for turbine maps. The scaling of pressure ratio, however, is slightly different:

$$\text{turbine PR scale factor} = \frac{PR_{\text{desired}} - 1}{PR_{\text{scaled}} - 1} = \frac{4.00 - 1}{3.58 - 1} = 1.163$$

In this example the desired turbine pressure ratio (the unscaled pressure ratio used to look up the turbine corrected flow and efficiency) is 4.00 while the actual turbine pressure ratio is 3.58, resulting in a scale factor of 1.163. NPSS will scale maps by any factor; it is up to the user to determine the validity of the resulting map. Certainly a map representing a single-stage axial fan should not be scaled to a design pressure ratio of 10.

Lastly, some maps have an additional input variable such as inlet guide vane (IGV) setting or nozzle vane angle. In this case there are multiple plots similar to figure 2 or figure 6 that comprise the turbomachine map, one such plot for each value of the additional variable. These are referred to as “stacked maps” or “3–D maps” and the third-dimensional variable is named “alpha” for compressors and ‘parmGeom’ for turbines.

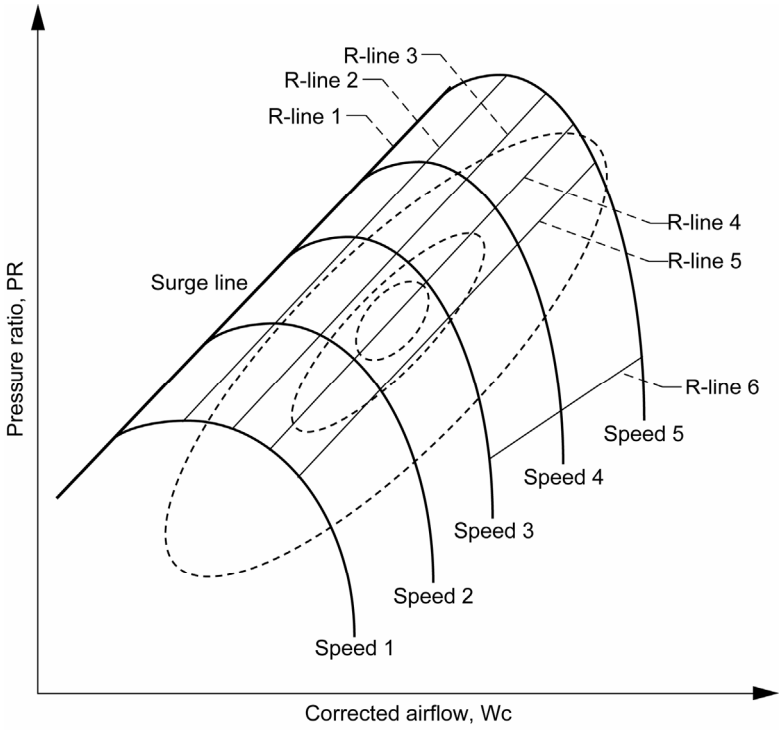

Figure 5.—Compressor map with R-lines.

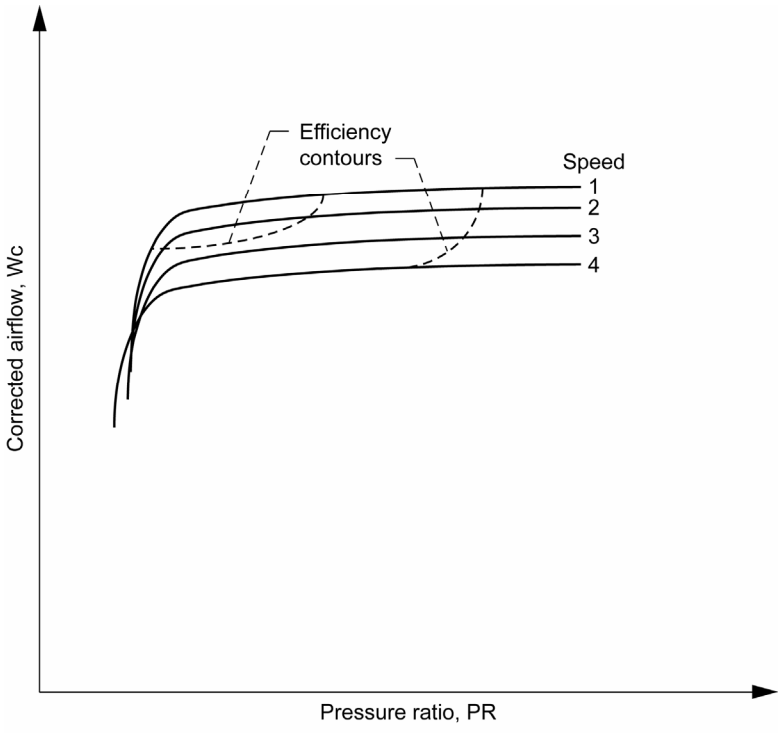

Figure 6.—Turbine map characteristics.

References

1. Lytle, J.K., The Numerical Propulsion System Simulation: An Overview, NASA/TM-2000-209915.
   NPSS User Guide Software Release: NPSS_1.6.3 AL
   NPSS Reference Sheets Software Release: NPSS_1.6.3 AL
2. Gordon, S., McBride, B.J., Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications, NASA RP–1311, 1996.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
28-03-2007	Technical Memorandum	

4. TITLE AND SUBTITLE
An Introduction to Thermodynamic Performance Analysis of Aircraft Gas Turbine Engine Cycles Using the Numerical Propulsion System Simulation Code

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Jones, Scott, M.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
WBS 984754.02.07.03.12.02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER
E-15876

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORS ACRONYM(S)
NASA

11. SPONSORING/MONITORING REPORT NUMBER
NASA/TM-2007-214690

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Categories: 07 and 61
Available electronically at http://gltrs.grc.nasa.gov
This publication is available from the NASA Center for AeroSpace Information, 301-621-0390

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This document is intended as an introduction to the analysis of gas turbine engine cycles using the Numerical Propulsion System Simulation (NPSS) code. It is assumed that the analyst has a firm understanding of fluid flow, gas dynamics, thermodynamics, and turbomachinery theory. The purpose of this paper is to provide for the novice the information necessary to begin cycle analysis using NPSS. This paper and the annotated example serve as a starting point and by no means cover the entire range of information and experience necessary for engine performance simulation. NPSS syntax is presented but for a more detailed explanation of the code the user is referred to the NPSS User Guide and Reference document (ref. 1)

15. SUBJECT TERMS
Aircraft engines; Gas turbine engines; Thermodynamic cycles; Jet propulsion; Object-oriented programming; Systems simulation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Scott M. Jones
U	U	U		47	19b. TELEPHONE NUMBER *(include area code)* 216-977-7015